Unicode im BS2000/OSD

Übersichtshandbuch

Ausgabe März 2007

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@fujitsu-siemens.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen. Zudem ist Unicode – anders als frühere Codierungssysteme, wie beispielsweise 7- oder 8-Bit-Codierungen – unabhängig von unterschiedlichen Herstellern, Systemen und Ländern.

Im Zuge der zunehmenden Internationalisierung von Software-Programmen und da immer mehr Kunden ihre BS2000/OSD-Anwendungen dem Internet öffnen, das Unicode-Codierungen verwendet, kommt dem Zeichensatz Unicode auch innerhalb des BS2000/OSD und seiner Anwendungen eine entscheidende Bedeutung zu.

1.1 Zielgruppe

Dieses Übersichtshandbuch richtet sich an Anwendungsprogrammierer und Systemverwalter, die sich einen Überblick verschaffen wollen, welche Unicode-Unterstützung ihnen im BS2000/OSD geboten wird, und welche BS2000-Komponenten sie dazu benötigen.

Es ersetzt nicht die Handbücher zu den einzelnen Produkten.

1.2 Konzept des Handbuchs

Kapitel 2 gibt einen kurzen Überblick über Unicode und die Motivation für die Unicode-Unterstützung im BS2000/OSD.

Kapitel 3 und 4 liefern grundlegende Definitionen und Konzepte zu Unicode, die für alle von Unicode betroffenen BS2000/OSD-Produkte gelten, und deren Handbücher jeweils ergänzen.

Darüber hinaus geben Kapitel 5 und 6 einen Überblick, wie die Unicode-Unterstützung in einzelnen, wesentlichen Anwendungen und Dateiverarbeitungsprogrammen im BS2000/OSD ausgestaltet ist.

Ergänzend bietet Kapitel 7 Tipps und Tricks für die Umstellung von BS2000/OSD-Anwendungen auf Unicode.

Im Anhang finden Sie hilfreiche Tabellen aus dem Umfeld der Unicode-Konvertierung.

Fachwort-, Abkürzungs-, Tabellen-, Literatur- und Stichwortverzeichnis runden das Handbuch ab.

2 Überblick

2.1 Was ist Unicode

Unicode ist ein von der International Standardisation Organisation (ISO) und dem Unicode-Konsortium genormter alphanumerischer Zeichensatz zur Codierung von Zeichen, Buchstaben, Ziffern, Satzzeichen, Silbenzeichen, Sonderzeichen sowie Ideogrammen. Ein Ideogramm ist ein Schriftzeichen, das einen ganzen Begriff darstellt und dabei symbolische Zeichen für abstrakte Begriffe verwendet oder sich aus zwei oder mehreren Piktogrammen zusammensetzt. Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen.

Unicode ist umfassend

Unicode enthält somit neben den Buchstaben des lateinischen Alphabets mit all seinen landesspezifischen Besonderheiten auch das griechische, kyrillische, arabische, hebräische und thailändische Alphabet sowie die so genannten CJK-Schriften – die verschiedenen chinesischen, japanischen und koreanischen Schriften. Zusätzlich sind mathematische, kaufmännische und technische Sonderzeichen codiert. Damit ist Unicode unabhängig von Sprachen und Schriftarten und unterstützt somit die Internationalisierung, d.h. die Entwicklung von Software-Programmen, die sich leicht an alle möglichen Sprachen und Kulturen anpassen lassen. Umgekehrt ermöglicht Unicode auch die Lokalisierung, d.h. die Anpassung von Software an "lokale" sprachliche und kulturelle Gegebenheiten, ohne Programmänderung: Dazu können landesspezifische Eigenschaften eingestellt werden und Zeichen, die mit Grundzeichen verknüpft sind, einheitlich dargestellt werden (Normalisierung von diakritischen Zeichen).

Mehr-Byte-Zeichensätze

Vor Unicode gab es überwiegend Codierungssysteme, in denen Zeichen mit Hilfe von 7 Bits oder 8 Bits dargestellt werden, was die Zeichenmenge der zugehörigen Zeichensätze auf 128 bzw. 256 Zeichen beschränkt. Zu den Bekanntesten gehören ASCII und EBCDIC, die jeweils in unterschiedlichen nationalen und teils auch herstellerspezifischen Ausprägungen existieren.

Unicode ist demgegenüber hersteller- und systemunabhängig. Es verwendet Zeichensätze (Code Sets) der Länge 2 bzw. 4 Bytes für die Codierung jedes Textzeichens, die bei ISO als UCS-2 (Universal Character Set 2) bzw. UCS-4 bezeichnet werden. Bei UCS-2 entsprechen die ersten 256 der maximal 65.536 Zeichen den Zeichen des Zeichensatzes ISO Latin-1 (ISO 8859-1). ISO Latin-1 ist weit verbreitet und fasst die Zeichen der westeuropäischen Sprachen zusammen.

UTF-8 und UTF-16

Statt der durch ISO definierten Bezeichnung UCS-2 wird häufig die Bezeichnung UTF-16 (UCS Transformation Format 16 Bit) verwendet, ein vom Unicode-Konsortium definierter Standard. UTF-16 verwendet auch Zeichen der Länge 4 Bytes (Surrogates). Damit erweitert sich der Zeichenumfang auf über 1,1 Millionen Zeichen.

Neben der Nutzung von UTF-16 ist auch der Einsatz von UTF-8 (UCS Transformation Format 8 Bit) weit verbreitet. UTF-8 stellt jedes Unicode-Zeichen abhängig von seiner Position in der Länge 1 bis 4 Bytes dar. UTF-8 stimmt in den ersten 128 Zeichen mit ASCII überein.

Detaillierte Informationen entnehmen Sie u.a. der Internetseite des Unicode-Konsortiums: *http://www.unicode.org/standard/translations/german.html.*

2.2 Motivation für die Unicode-Unterstützung

Der Standardzeichensatz im BS2000/OSD ist EBCDIC.DF.03IRV (EDF03IRV), ein 7-Bit-Zeichensatz, dessen Zeichenvorrat dem ASCII-7-Bit-Zeichensatz entspricht, erweitert um den zweiten Steuerzeichenblock von ISO8859-1. EDF03IRV umfasst nur 95 abdruckbare Zeichen.

Eine Erweiterung auf 181 abdruckbare Zeichen erhält man durch die Einführung eines 8-Bit-Zeichensatzes. Diese Erweiterung erlaubt es, Zeichensätze für bestimmte Sprachräume aufzustellen. Sie reicht jedoch nicht aus, um die unterschiedlichen Zeichen aller europäischen Sprachen geschweige denn aller aktiven Sprachen in einem Ein-Byte-Zeichensatz abzubilden. Die im BS2000 bekannten 8-Bit-EBCDIC-Tabellen lehnen sich an den Zeichenvorrat der entsprechenden ISO8859-Tabellen an, wenn auch nicht jede ISO8859-n-Tabelle ihre Entsprechung im BS2000/OSD hat.

Anwendungen im Internet

Immer mehr BS2000/OSD-Kunden öffnen ihre BS2000/OSD-Anwendungen dem Internet. Der Zeichensatz im Internet ist UTF-8. Die Sprache Java nutzt UTF-16 zur Codierung der Daten. Über JAVA-Database-Connectivity (JDBC) sind JAVA-Anwendungen und JAVA-Applets mit BS2000/OSD-Datenbanken verbunden. Es ist abzusehen, dass eine Konvertierung der Daten aus dem Internet in eine Ein-Byte-Codierung nicht mehr ausreicht. Zudem ist eine buchstabengetreue Übertragung von Wörtern aus einem nicht-lateinischen Text in die lateinische Schrift (Transliteration) nicht erwünscht bzw. nicht möglich, da die entsprechenden diakritischen Zeichen im eingestellten Zeichensatz nicht vorhanden sind.

Europäische Sprachen

Wenn neben westeuropäischen auch osteuropäische Adressdaten in einer Spalte einer Datenbanktabelle gespeichert werden sollen, reicht der Zeichenvorrat einer EBCDIC-Tabelle nicht mehr aus. Denn die internationalen Regeln der Post für die Angabe einer Adresse besagen, dass der Absender Name und Anschrift des Empfängers in der Sprache des Bestimmungslandes und mit dessen Schriftzeichen anzugeben hat. Ebenso muss der Bestimmungsort in Auslandsanschriften in Großbuchstaben in der Schreibweise des Bestimmungslandes übermittelt werden. D.h. der Zeichensatz der Spalten "Name", "Wohnort", "Strasse" der Datenbanktabelle muss lateinische, griechische, kyrillische usw. Zeichen aufnehmen können.

3 Unicode-Codierungen

In Unicode wird jedem Zeichen eine Nummer, der so genannte Code Point, zugeordnet.

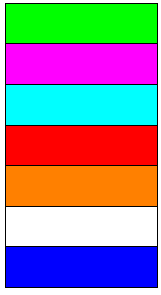
Ein Unicode Code Point wird im Allgemeinen in der Form U+n angegeben, wobei n aus 4 bis 6 hexadezimalen Ziffern besteht.

Die Menge aller Code Points bildet den so genannten Code Space. Der Code Space der Unicode-Norm V4 umfasst 1.114.112 Code Points, von denen die meisten noch nicht vergeben sind. Unterteilt ist der Code Space in die so genannten Ebenen. Jede Ebene umfasst 65.536 Code Points. Die wichtigste Ebene ist die Ebene 0, die Basic Multilanguage Plane (BMP), die die Code Points von U+0000 bis U+FFFF aufspannt. Die Code Points von U+0000 bis U+00FF stimmen mit denen von ISO8859-1 überein. Die Ebenen 15 und 16 reserviert die Unicode-Norm für die Definition privater Zeichen (Private Use Area).

Ebene		– 0FFF	– 1FFF	– 2FFF	– 3FFF	– 4FFF	– 5FFF	– 6FFF	– 7FFF	– 8FFF	– 9FFF	– AFFF	– BFFF	– CFFF	– DFFF	– EFFF	– FFFF
0	00000																
1	10000																
2	20000																
3	30000																
4	40000																
5	50000																
6	60000																
7	70000																
8	80000																
9	90000																
10	A0000																
11	B0000																
12	C0000																
13	D0000																
14	E0000																
15	F0000																
16	100000																

Tabelle 1: Code Space der Unicode-Norm V4

Legende:



General Scripts, Symbols
Chinesiche, japanische und koreanische Schriften (CJK)
Surrogates
Private Use Area
Zusammengesetzte Zeichen
nicht vergebene Code Points (unused)
Tag characters

Bei der Unterstützung von Unicode ist es von wesentlicher Bedeutung, in welcher Weise diese Code Points in Bytes codiert werden. Das Unicode-Konsortium definiert dafür drei verschiedene Codierungsmöglichkeiten: UTF-8, UTF-16 und UTF-32.

Ein Sonderfall, der nicht direkt im Unicode-Standard steht, ist UTF-EBCDIC, eine Unicode-Codierung für Server, die den EBCDIC-Zeichensatz verwenden.

Die folgenden Abschnitte liefern detaillierte Informationen zu den verschiedenen Codierungsmöglichkeiten, sowie zu Normalisierung und Sortierung von Unicode-Zeichenketten.

3.1 UTF-8

UTF-8 verwendet eine variable Anzahl von Bytes zur Codierung der Unicode-Zeichen. Die Byte-Darstellung der ASCII-Zeichen bleibt unverändert. Für alle weiteren Zeichen gibt die Anzahl der führenden Einsen im ersten Byte an, wie viele Bytes zu einem Zeichen gehören. Folge-Bytes fangen immer mit 10 an.

UTF-8	**Serialisierte Bytes**			
Unicode-Bereich	**1. Byte**	**2. Byte**	**3. Byte**	**4. Byte**
U+000000 - U+00007F	***0***nnnnnnn			
U+000080 - U+0007FF	***110***nnnnn	***10***nnnnnn		
U+000800 - U+00FFFF	***1110***nnnn	***10***nnnnnn	***10***nnnnnn	
U+010000 - U+10FFFF	***11110***nnn	***10***nnnnnn	***10***nnnnnn	***10***nnnnnn

Beispiel

Unicode-Zeichen	**UTF-8**
U+ 20AC (Euro-Zeichen)	***1110***0010 ***10***000010 ***10***101100

UTF-8 hat den Vorteil gegenüber älteren Multi-Bytes-Zeichensätzen, dass bei einem Zeichen, das mit mehreren Bytes codiert ist, keines der Einzel-Bytes ein gültiges Zeichen repräsentiert.

Da UTF-8 ASCII-kompatibel ist, ändert sich der Speicherbedarf für Zeichenketten bei dieser Art der Unicode-Codierung im Englischen nicht. Für alle anderen Latin-x-Zeichensätze werden die Texte im Durchschnitt 10% länger. Griechische, kyrillische oder arabische Zeichen liegen im Bereich zwischen 128 und 2047 (U+00000080 bis U+000007FF), deshalb wird der Speicherbedarf für Texte in diesen Sprachen etwa 70% größer.

Die Zeichen der ostasiatischen Sprachen liegen im Bereich darüber und benötigen pro Zeichen etwa 3 Bytes.

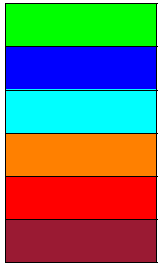
Für eine Norm-konforme UTF-8-Codierung gilt immer die kürzeste Codierung, d.h. der Code Point U+7F ist immer als 0111 1111 = x'7F' darzustellen und nicht als Zwei-Bytes-Codierung 1100 0001 1011 1111 = x'C1BF'.

Aus dieser Regel ergibt sich die folgende Tabelle gültiger UTF-8-Byte-Belegungen.

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	00	10	20	30	40	50	60	70						400-43F	800-FFF	1000-3FFF
-1	01	11	21	31	41	51	61	71						440-47F	1000-1FFF	4000-7FFF
-2	02	12	22	32	42	52	62	72					80-BF	480-4BF	2000-2FFF	8000-BFFF
-3	03	13	23	33	43	53	63	73					C0-FF	4C0-4FF	3000-3FFF	C000-FFFF
-4	04	14	24	34	44	54	64	74					100-13F	500-53F	4000-4FFF	10000-10FFFF
-5	05	15	25	35	45	55	65	75					140-17F	540-57F	5000-5FFF	
-6	06	16	26	36	46	56	66	76					180-1BF	580-5BF	6000-6FFF	
-7	07	17	27	37	47	57	67	77					1C0-1FF	5C0-5FF	7000-7FFF	
-8	08	18	28	38	48	58	68	78					200-23F	600-63F	8000-8FFF	
-9	09	19	29	39	49	59	69	79					240-27F	640-67F	9000-9FFF	
-A	0A	1A	2A	3A	4A	5A	6A	7A					280-2BF	680-6BF	A000-AFFF	
-B	0B	1B	2B	3B	4B	5B	6B	7B					2C0-2FF	6C0-6FF	B000-BFFF	
-C	0C	1C	2C	3C	4C	5C	6C	7C					300-33F	700-73F	C000-CFFF	
-D	0D	1D	2D	3D	4D	5D	6D	7D					340-37F	740-77F	D000-DFFF	
-E	0E	1E	2E	3E	4E	5E	6E	7E					380-3BF	780-7BF	E000-EFFF	
-F	0F	1F	2F	3F	4F	5F	6F	7F					3C0-3FF	7C0-7FF	F000-FFFF	

Tabelle 2: Byte-Belegung in der Darstellung von UTF-8

Legende:



Ein-Byte-Codierung (Nummer entspricht dem Unicode Code Point)

Folge-Byte einer Mehr-Bytes-Darstellung

Erstes Byte einer Zwei-Bytes-Codierung

Erstes Byte einer Drei-Bytes-Codierung

Außerhalb des Code Spaces von Unicode V4.0 (Keine gültige UTF-8-Codierung)

Erstes Byte einer Vier-Bytes-Codierung (außerhalb der BMP)

3.2 UTF-EBCDIC (UTFE)

Für Systeme, die EBCDIC verwenden, existiert beim Unicode-Konsortium ein Technischer Report, der eine Konvertierung von Unicode-Zeichen in EBCDIC vorschlägt:

Das dort beschriebene Verfahren erweitert in einem ersten Schritt den Bereich der Ein-Byte-Codierung der Unicode-Zeichen von UTF-8 um den zweiten Steuerzeichenblock (U+80 bis U+9F) und schränkt den Bereich der Folge-Bytes einer Mehr-Bytes-Codierung auf den Bereich x'A0' bis x'BF' ein. Die sich ergebende Codierung wird modifiziertes UTF-8 (UTF-8MOD) genannt.

UTF-8MOD	**Serialisierte Bytes**				
Unicode-Bereich	**1. Byte**	**2. Byte**	**3. Byte**	**4. Byte**	**5. Byte**
U+000000 - U+00007F	***0***nnnnnnn				
U+000080 - U+00009F	***100***nnnnn				
U+0000A0 - U+0003FF	***110***nnnnn	***101***nnnnn			
U+000400 - U+003FFF	***1110***nnnn	***101***nnnnn	***101***nnnnn		
U+004000 - U+03FFFF	***11110***nnn	***101***nnnnn	***101***nnnnn	***101***nnnnn	
U+040000 - U+10FFFF	***111110***0n	***101***nnnnn	***101***nnnnn	***101***nnnnn	***101***nnnnn

Im zweiten Schritt wird mit Hilfe der Standardkonvertierung von ISO8859-n nach EBCDIC.DF.04.n das modifizierte UTF-8 zu UTF-EBCDIC (UTFE) umgewandelt.

Beispiel

Unicode-Zeichen	UTF-8MOD	UTF-EBCDIC
U+ 20AC (Euro-Zeichen)	***1110***1000 ***101***00101 ***101***01100 E8 A5 AC	80 B2 BA

Der sich so ergebende Ein-Byte-Bereich der UTFE-Codierung (siehe Tabelle 4 auf Seite 21) entspricht dem Zeichensatz EBCDIC.DF.03IRV (EDF03IRV), dem Standardzeichensatz im BS2000/OSD. Im Gegensatz zu den anderen Unicode-Codierungen, kann UTFE von allen BS2000/OSD-Anwendungen geparst werden, die sich auf EDF03IRV beschränken. Dies gilt insbesondere für das BS2000/OSD selbst. Ein BS2000/OSD-Kommando enthält immer nur Zeichen aus dem Ein-Byte-Bereich von UTFE. Wenn ein Zeichen aus dem Mehr-Bytes-Bereich in der Eingabezeichenkette enthalten ist, wird korrekt auf Syntaxfehler erkannt. Auch alle Steuerzeichen befinden sich im Ein-Byte-Bereich.

Diese Umsetzung weicht von dem Vorschlag des technischen Reports der Unicode-Norm ab, da die EBCDIC-Codierung im BS2000/OSD und im z/OS nicht identisch ist. D.h. die UTFE-Codierung in ORACLE entspricht nicht der im BS2000/OSD.

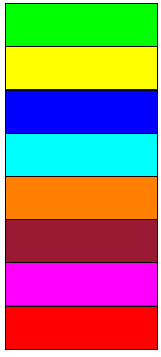
Da jedoch beide auf der UTF-8MOD-Codierung basieren, kann durch eine geeignete Umsetztabelle die eine Form leicht in die andere überführt werden.

Für eine Norm-konforme UTF-8MOD-Codierung gilt immer die kürzeste Codierung, d.h. der Code Point U+7F darf nach obigen Schema nur durch 0111 1111 = x'7F' dargestellt werden, nicht durch die Zwei-Bytes-Codierung 1100 0011 1011 1111= x'C3BF'. Aus dieser Regel ergibt sich die nachfolgende Tabelle 3 mit den gültigen UTF-8MOD-Byte-Belegungen als Vorstufe zu UTFE.

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	00	10	20	30	40	50	60	70	80	90				200-21F		4000-7FFF
-1	01	11	21	31	41	51	61	71	81	91				220-23F	400-7FF	8000-FFFF
-2	02	12	22	32	42	52	62	72	82	92				240-25F	800-BFF	10000-17FFF
-3	03	13	23	33	43	53	63	73	83	93				260-27F	C00-FFF	18000-1FFFF
-4	04	14	24	34	44	54	64	74	84	94				280-29F	1000-13FF	20000-27FFF
-5	05	15	25	35	45	55	65	75	85	95			A0-BF	2A0-2BF	1400-17FF	30000-37FFF
-6	06	16	26	36	46	56	66	76	86	96			C0-DF	2C0-2DF	1800-1BFF	38000-3FFFF
-7	07	17	27	37	47	57	67	77	87	97			E0-FF	2E0-2FF	1C00-1FFF	40000-FFFFF
-8	08	18	28	38	48	58	68	78	88	98			100-11F	300-31F	2000-23FF	100000-10FFFF
-9	09	19	29	39	49	59	69	79	89	99			120-13F	320-33F	2400-27FF	
-A	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A			140-15F	340-35F	2800-2BFF	
-B	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B			160-17F	360-37F	2C00-2FFF	
-C	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C			180-19F	380-39F	3000-33FF	
-D	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D			1A0-1BF	3A0-3BF	3400-37FF	
-E	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E			1C0-1DF	3C0-3DF	3800-3BFF	
-F	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F			1E0-1FF	3E0-3FF	3C00-3FFF	

Tabelle 3: Byte-Belegung in der Darstellung von UTF-8MOD

Legende für Tabelle 3 und Tabelle 4:



Ein-Byte-Codierung
Zweiter Steuerzeichenblock: Ein-Byte-Codierung
Folge-Byte einer Mehr-Bytes-Darstellung
Erstes Byte einer Zwei-Bytes-Codierung
Erstes Byte einer Drei-Bytes-Codierung
Erstes Byte einer Vier-Bytes-Codierung
Erstes Byte einer Fünf-Bytes-Codierung
Keine gültige UTF-8MOD-Codierung

Die nachfolgende Tabelle 4 stellt die Byte-Belegung in der Darstellung von UTF-EBCDIC (UTFE) dar. Diese Byte-Belegung ist das Ergebnis der Umsetzung von UTF-8MOD nach UTFE mit Hilfe der Standardkonvertierung von ISO8859-n nach EBCDIC.DF.04.n.

Der Ein-Byte-Bereich dieser UTFE-Codierung entspricht dem Standardzeichensatz im BS2000/OSD (EBCDIC.DF.03IRV).

	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	00	01	02	03	85	09	86	7F	87	8D	8E	0B	0C	0D	0E	0F
-1	10	11	12	13	8F	92	08	97	18	19	9C	9D	1C	1D	1E	1F
-2	80	81	82	83	84	0A	17	1B	88	89	8A	8B	8C	05	06	07
-3	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
-4	20	A0	800-BFF	1000-13FF	E0	400-7FF	C00-FFF	1400-17FF	1C00-1FFF	8000-FFFF	60	2E	3C	28	2B	7C
-5	26	2400-27FF	2800-2BFF	2C00-2FFF	2000-23FF	3400-37FF	3800-3BFF	3C00-3FFF	3000-33FF	3E0-3FF	21	24	2A	29	3B	9F
-6	2D	2F	C2	C4	C0	C1	C3	A0-BF	E0-FF	220-23F	5E	2C	25	5F	3E	3F
-7	100000 - 10FFFF	120-13F	140-15F	160-17F	100-11F	1A0-1BF	1C0-1DF	1E0-1FF	180-19F	A8	3A	23	40	27	3D	22
-8	300-31F	61	62	63	64	65	66	67	68	69	AB	BB	4000-7FFF	FD	FE	B1
-9	B0	6A	6B	6C	6D	6E	6F	70	71	72	AA	BA	1800-1BFF	B8	C0-DF	A4
-A	B5	AF	73	74	75	76	77	78	79	7A	A1	BF	200-21F	3A0-3BF	3C0-3BF	AE
-B	A2	A3	A5	B7	A9	A7	B6	BC	BD	BE	AC	5B	5C	5D	B4	2E0-2FF
-C	F9	41	42	43	44	45	46	47	48	49	AD	20000 - 27FFF	38000 - 3FFFF	10000 - 17FFF	18000 - 1FFFF	30000 - 37FFF
-D	A6	4A	4B	4C	4D	4E	4F	50	51	52	B9	FB	FC	360-37F	FA	FF
-E	320-33F	40000-FFFFF	53	54	55	56	57	58	59	5A	B2	D4	D6	D2	D3	D5
-F	30	1F	32	33	34	35	36	37	38	39	B3	7B	380-39F	7D	340-35F	7E

Tabelle 4: Byte-Belegung in der Darstellung von UTFE

3.3 UTF-16

Bei UTF-16 werden alle Unicode-Zeichen zwischen U+0000 und U+FFFF durch 2 Bytes codiert. Die Zeichen in diesem Bereich werden häufig auch als Universal Character Set der Länge 2 (UCS-2) bezeichnet. Alle Zeichen oberhalb von U+FFFF werden durch 4 Bytes dargestellt, so genannte Surrogate Pairs.

Um diese Surrogate Pairs eindeutig zu kennzeichnen, wurde der Bereich zwischen U+D800 und U+DFFF für diese Paare reserviert. Die ersten 16 Bits eines solchen Paares fangen immer mit x'110110' an und die zweiten mit x'110111'. Die restlichen Bits des darzustellenden Zeichens werden ähnlich wie bei UTF-8 auf die übrigen Stellen verteilt.

Zeichen, die mit UTF-16 codiert sind, haben also wie in UTF-8 keine feste Länge. In der Praxis reichen aber 2 Bytes aus. Microsoft Windows und JAVA beispielsweise unterstützen Unicode als Zeichen mit 2 Byte Länge, also UTF-16 ohne Surrogate Pairs (UCS-2).

Bei dieser Codierung muss zusätzlich beachtet werden, wie der Prozessor die Bytes anordnet: Bei Big Endian liegt das wichtigste Byte an der niedrigsten Speicheradresse, bei Little Endian liegt das unwichtigste Byte an der niedrigsten Speicheradresse:

Little Endian wird beispielsweise von allen Intel-Systemen verwendet, Big Endian z.B. auf SPARC, von TCP/IP oder der Java Virtual Machine.

UTF-16 verdoppeln die Textlänge für alle gängigen nicht ostasiatischen Sprachen.

Beispiel

Unicode-Zeichen	Big Endian	Little Endian
U+ 20AC (Euro-Zeichen)	00100000 10101100	10101100 00100000

3.4 UTF-32

Jedes Zeichen des Unicode-Standards wird direkt als 32-Bits-Einheit codiert. Damit vervierfacht sich auch der Speicherbedarf für die europäischen Sprachen. Auch bei dieser Codierung muss die Prozessoreigenschaft Big Endian bzw. Little Endian beachtet werden, siehe auch Abschnitt „UTF-16“.

Wegen seines großen Speicherplatzbedarf spielt UTF-32 noch keine große Rolle.

3.5 Normalisierung

Die Codierung eines Zeichens in Unicode ist nicht eindeutig, d.h. es gibt für ein Zeichen unter Umständen mehr als eine Codierung.

Ein typisches Beispiel sind die deutschen Umlaute. Für das „Ä“ gibt es die folgenden Unicode-Verschlüsselungen: ’u+00C4’ und die Kombination aus „A“ (’u+0041’) und dem Umlautzeichen (’u+0308’).

Diese Eigenschaft von Unicode ist für die Programmierung sehr hinderlich.

Deshalb gibt es die Normalisierungsfunktionen DECOMPOSE und COMPOSE.

Die Funktion DECOMPOSE zerlegt jedes „zusammengesetzte“ Zeichen in seine einzelnen Bestandteile, in das Grundzeichen und in die damit verknüpften diakritischen Zeichen. Die Reihenfolge der verknüpften diakritischen Zeichen ist dabei streng festgelegt.

Die Funktion COMPOSE bildet alle Code Points, die zusammen ein Zeichen ergeben, in den entsprechenden Code Point ab.

Der Vorgang der Normalisierung benötigt eine sehr große Rechenleistung. Deshalb geht das BS2000/OSD, ebenso wie der SQL-Standard davon aus, dass die Daten in normalisierter, d.h. komprimierter Form vorliegen.

Wenn nicht sichergestellt ist, ob Daten in normalisierter Form vorliegen, sollte eine Normalisierung durchgeführt werden. Im BS2000/OSD werden die Normalisierungsfunktionen durch die Komponente XHCS angeboten, siehe auch XHCS (BS2000/OSD)-Handbuch. Wenn der Inhalt einer Datei in die „zusammengesetzte“ Form überführt werden soll, kann dies mit Hilfe des BS2000/OSD-Dienstprogramms PERCON geschehen, siehe auch PERCON V2.9A (BS2000/OSD)-Handbuch. Die Normalisierung im BS2000/OSD ist auf den Bereich der Code Points ’u+0000’ bis ’u+2FFF’ beschränkt.

Beispiel

Der Unicode Code Point ’u+1ED6’ entspricht dem lateinischen Großbuchstaben „O“ mit Zirkumflex und Tilde. Dieses Zeichen kann mit Hilfe dreier Unicode Code Points ’u+00D4’ für „Ô“ und ’u+0303’ für Tilde bzw. ’u+004F’ für „O“ und ’u+302’ für Zirkumflex und ’u+0303’ für Tilde erzeugt werden. Auch die Unicode Code Point-Sequenz ’u+00D5’ für das „Õ“ mit Tilde und ’u+302’ für Zirkumflex ergibt dieses Zeichen. Es gibt nur die Regel, dass das Grundzeichen vor den damit verknüpften diakritischen Zeichen steht.

D.h. das Ergebnis der DECOMPOSE-Funktion des lateinischen Großbuchstaben „O“ mit Zirkumflex und Tilde ist die Unicode Code Point-Folge ’u+004F’, ’u+0302’, ’u+0303’, das Ergebnis der COMPOSE-Funktion ist ’u+1ED6’.

3.6 Sortierreihenfolge

In den meisten Programmen werden Zeichenketten binär verglichen. Die binäre Sortierreihenfolge der Zeichen ist abhängig von ihrer Codierung:

- Für alle ISO8859- und Unicode-Codierungen gilt:
  Ziffern (1-9) < lateinische Großbuchstaben (A-Z) < lateinische Kleinbuchstaben (a-z)
- Für EBCDIC und UTFE gilt:
  lateinische Kleinbuchstaben (a-z) < lateinische Großbuchstaben (A-Z) < Ziffern (1-9).

i Wenn Sätze Daten sowohl im UTF-16-Code als auch im EBCDIC-Code enthalten, achten Sie darauf, dass nach dem richtigen Algorithmus sortiert wird.

Die Unicode-Norm beschreibt einen linguistischen Sortieralgorithmus, der Vergleiche auf mehreren Ebenen durchführt.

Hierzu wird jedem Zeichen ein Sortierelement (Collation Element) zugeordnet, dass das Gewicht des Zeichens innerhalb der einzelnen Vergleichsebene beschreibt. Innerhalb jeder Ebene ist die Reihenfolge durch Zahlen festgelegt.

Der sogenannte Sortierschlüssel (Sort Key) einer Zeichenkette wird gebildet, indem die Zahlen jeder Ebene zu jeweils einer Zeichenkette zusammengefasst werden. Wenn der Wert in einer Ebene eines Sortierelements binär Null ist, wird dieses Element in dieser Ebene nicht zum Aufbau des Sortierschlüssels herangezogen. Zwei Zeichenketten werden mit Hilfe des Sortierschlüssels verglichen. Diese werden Ebene für Ebene miteinander verglichen. Der erste Unterschied bestimmt das Vergleichsergebnis.

Im BS2000/OSD wird die Sortierung bis zur Ebene 3 unterstützt. Die Bedeutung der einzelnen Ebenen beschreibt die nachfolgende Tabelle.

Vergleichsebene	**Beschreibung**	**Beispiel**
Ebene 1	Grundzeichen (base character)	role < roles < rule
Ebene 2	Akzente	role < rôle < roles
Ebene 3	Groß-/Kleinschreibung	role < Role < rôle

Auf dem höchsten Niveau (Ebene 1) sind den Grundzeichen Wertigkeiten zugewiesen, ohne Einfluss der nachfolgenden Zeichen und diakritischer Zusätze. Auf Ebene 2 werden Grundzeichen mit diakritischen Zusätzen, z.B. Akzente, Tilde, unterschieden. Wenn der gesamte Sortierbegriff auf Ebene 1 gleich ist, kommen Unterschiede auf Ebene 2 zum Tragen. Wenn Ebene 1 und Ebene 2 keinen Unterschied liefern, werden auf Ebene 3 zusätzlich Großbuchstaben und Kleinbuchstaben unterschieden. Verglichen wird immer von links nach rechts.

Die Werte für die Sortierelemente (Unicode Default Collation Table), die unter der Web-Seite des Unicode-Konsortiums veröffentlicht sind, können sich ändern. Diese Tabelle finden Sie z.B. unter der Adresse:
http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt.

Im BS2000/OSD können Sie sich das Sortierelement über XHCS besorgen, siehe auch XHCS (BS2000/OSD)-Handbuch.

4 Unicode im BS2000/OSD

4.1 Grundlagen der Unicode-Unterstützung im BS2000/OSD

Einsatzszenario

Mit der Unicode-Unterstützung im BS2000/OSD werden die in BS2000/OSD-Systemen verfügbaren Code Sets um zusätzliche Zeichen erweitert. Die Programmier- und Ablauf-Umgebung, die Sie benötigen, um Ihre bestehenden Anwendungen um Unicode-Datenfelder erweitern zu können, wird zur Verfügung gestellt. Es wird davon ausgegangen, dass die Anzahl der Felder, die auf Unicode umgestellt oder die zusätzlich eingefügt werden müssen, gering ist. Im Wesentlichen handelt es sich dabei um Namens- und Adressfelder.

Sie müssen nur solche Anwendungen oder Anwendungsteile modifizieren, die auch die erweiterte Funktionalität nutzen.

Unicode wird im BS2000/OSD auf Basis der vorhandenen Produkte unterstützt. Unicode-basierte Zeichen werden nur für die zu verarbeitenden oder zu verwaltenden Texte zugelassen, d.h. für die Feld- bzw. Containerinhalte, nicht jedoch für Feld- bzw. Containernamen. Die produktspezifischen Regeln für Kommandos und Objektnamen bleiben bestehen.

Konzept des Coded Character Set Name (CCSN)

Zur Unterstützung verschiedener Zeichensätze und Codierungen dient das Konzept der Coded Character Sets (CCS). Ein CCS definiert einen Zeichensatz und die Codierung dieser Zeichen in der Datei. XHCS ist die zentrale Informationsquelle für die CCS, die im BS2000/OSD zur Verfügung stehen, siehe auch Seite 36. Der CCS-Name dient zur Identifikation der verschiedenen Zeichensätze und Codierungen.

Während der Umstellung von einem 7-Bit auf einen 8-Bit-Zeichensatz und danach ist es notwendig, umgestellte Daten von nicht umgestellten Daten zu unterscheiden.

Deshalb können Sie in der BS2000/OSD-BC V7.0 den Standardwert des CCS-Namens einer Datei wie folgt beeinflussen:

- Grundsätzlich gilt, dass eine explizite Angabe des CCS-Namen immer Vorrang hat.
- Wenn eine Datei neu angelegt wird, wird der CCS-Name des Benutzereintrages des aufnehmenden Public Volume Set als CCS-Name der Datei übernommen, falls dieser nicht EDF03IRV ist.
  Wenn er EDF03IRV ist, erhält die Datei - wie bisher - den CCS-Namen *NONE. Analog verhält sich LMS bezüglich neuer Elemente; diese erhalten den CCS-Namen der Bibliothek, falls kein Wert für den CCS-Namen angegeben ist.
- Wenn Sie mit dem System-Standard-Zeichensatz arbeiten, ändert sich nichts gegenüber dem bisherigen Verhalten.
- Wenn Sie mit einem 8-Bit-Zeichensatz arbeiten, ist die Nutzung eines Codes auf Dateiebene und Bibliotheks-Elementebene klar spezifiziert.
- Wenn Sie eine Datei kopieren, sichern oder rücksichern, wird das Dateiattribut CCS-Name immer mit transportiert.

Eine Übersicht über die CCSN-Standardwerte entnehmen Sie der Tabelle „Standardwerte für CCSN“ auf Seite 63.

Weitere Grundsätze der Unicode-Einbettung im BS2000/OSD

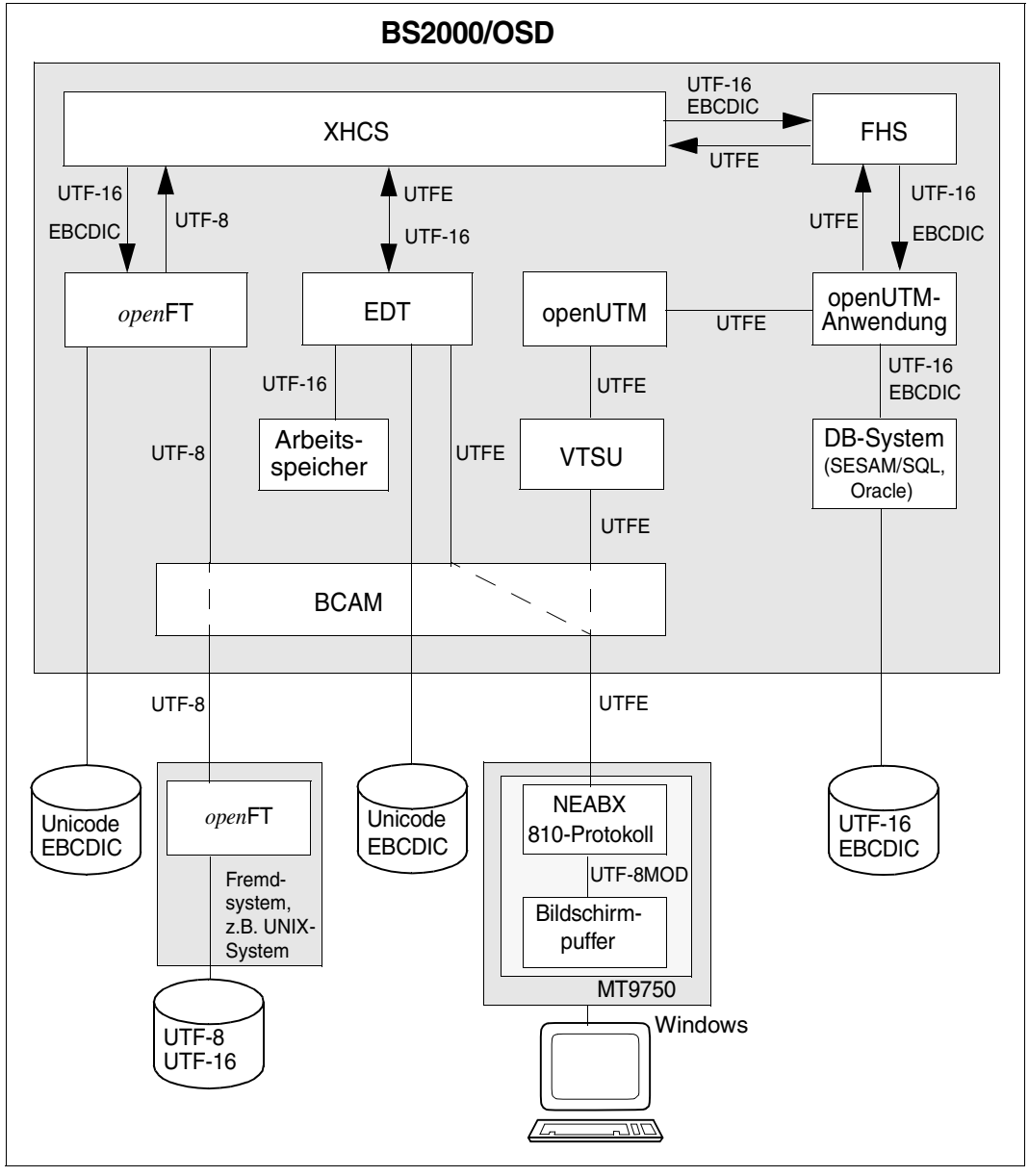
Darüber hinaus wurden folgende Überlegungen bei der Einbettung von Unicode im BS2000/OSD zugrunde gelegt:

- Innerhalb von Anwendungen werden Unicode-Zeichen in UTF-16 codiert, auf Kommandoebene in UTFE.
- Der Unicode-Zeichenvorrat ist auf UCS-2 begrenzt, d.h. Surrogate Pairs (siehe auch Abschnitt „UTF-16“ auf Seite 22) werden nicht unterstützt.
- XHCS bietet die Dienste und das Interface für Codierungsaufgaben und stellt die Konvertierungen zwischen den Codierungen (EBCDIC, Unicode) zur Verfügung.
- Als Einsatzgebiet wird Zentraleuropa angenommen:
  - Konvertierungsfunktionen werden nur aus dem europäischen Sprachraum angeboten.
  - Sortierung und Normalisierung stehen nur bis zum Unicode-Codepoint U+2FFF zur Verfügung. Ab U+3000 beginnen die chinesischen, japanischen und koreanischen Zeichen.
  - Es stehen nur Glyphen für europäische Sprachen zur Verfügung.

- Die Größe aller BS2000/OSD- und Datenbank-Container bleibt erhalten.
  Da ein UTF-16-Zeichen zwei Bytes Speicherplatz belegt, können daher beispielsweise in einem SAM-Satz abhängig von der Dateiorganisation maximal etwa 16.000 Zeichen und nicht wie bisher etwa 32.000 Zeichen gespeichert werden.
- Privatplatten werden nicht unterstützt, da diese das Abspeichern des CCSN nicht zulassen.

4.2 Überblick über die betroffenen Schnittstellen

Die nachfolgende Abbildung stellt Unicode im BS2000/OSD-Systemumfeld beispielhaft in Form der Unicode-Codierungen an den einzelnen Schnittstellen dar. Für eine größere Übersichtlichkeit wurde auf einige technische Details verzichtet.



Legende:

EBCDIC	beliebige EBCDIC-Codierung z.B. EDF03IRV, EDF041 usw.
Unicode	mögliche Unicode-Codierungen UTF-8, UTFE, UTF-16

Die Unicode-Codierungen an den unterschiedlichen Schnittstellen im BS2000/OSD ergeben sich aus den unterschiedlichen Anforderungen: UTF-8 ist die bevorzugte Unicode-Codierung in Netzen. Deshalb nutzt das Dateiübertragungsprogramm *open*FT die Codierung UTF-8 zum Transport von Daten in heterogenen Umgebungen. D.h. *open*FT konvertiert die zu sendenden Daten nach UTF-8 und überträgt sie im nicht-transparenten Modus an den *open*FT-Partner im Zielsystem, der wiederum die Daten in das gewünschte Zielformat konvertiert. Für die Konvertierung im BS2000/OSD nutzt *open*FT die Systemkomponente eXtended Host Code Support (XHCS).

UTFE entspricht in seiner Ein-Byte-Codierung EDF03IRV, d.h. Parser, deren Syntaxelemente keine Zeichen außerhalb des Zeichenvorrats EDF03IRV nutzen, können auch eine UTFE-Zeichenkette verarbeiten. Ausgenutzt wird diese Eigenschaft auf der BS2000-Kommandoebene und z.B. im Editor zur Dateiaufbereitung (EDT) beim Parsen der EDT-Anweisung.

Die folgende Sequenz kann im BS2000/OSD korrekt abgearbeitet werden:

```
/MODIFY-TERMIMAL-OPTIONS CODED-CHARACTER-SET=UTFE              (1)
...
/START-EDTU                                                   (2)
@COPY F=UNICODE-FILE                                          (3)
@ON&CA 'Dolina Kukol' TO 'Долина Кукол'                       (4)
...
```

Erläuterung:

(1) schaltet die Eingabe auf UTFE um.

(2) ruft den EDT auf. Die UTFE-Codierung unterscheidet sich nicht von EDF03IRV, d.h. der BS2000-Kommandoprozessor kann das Kommando korrekt interpretieren.

(3) öffnet die Datei UNICODE-FILE und liest sie in den Arbeitspeicher ein. Dateinamen, die nicht den BS2000-Dateinamenskonventionen entsprechen, werden abgelehnt. Beim Einlesen der Daten aus der Datei in den Arbeitsspeicher werden die Daten nach UTF-16 konvertiert. Dabei wird der Coded Character Set Name (CCSN) der Datei ausgewertet. Für die interne Verarbeitung von Unicode-Daten verwendet der EDT die Unicode-Codierung UTF-16, da diese Codierung in dem von BS2000/OSD unterstützten Umfang (UCS-2) eine Codierung mit fester Länge ist, die jedem Zeichen genau 2 Bytes zuordnet. Diese eignet sich besser für die Verarbeitung als die in der Länge variable Unicode-Codierung UTFE.

(4) Auch das Ersetzen-Kommando erkennt der EDT-Kommando-Interpreter. Für die Durchführung des Kommandos werden jedoch die beiden Strings 'Dolina Kukol' und ’Долина Кукол’ von UTFE nach UTF-16 konvertiert. Verarbeitet werden die Daten anschließend in UTF-16.

Für VTSU ist UTFE nur ein weiterer 8-Bit EBCDIC-Zeichensatz. Alle VTSU-Steuerzeichen haben die gleiche Codierung wie in EDF03IRV. Bei der Terminaleinstellung UTFE werden an der RDATA-Schnittstelle nur die Zeichen von a-z nach A-Z standardmäßig von Klein- auf Großschreibung umgesetzt.

FHS bietet mit der Version 8.3A die Möglichkeit, sowohl UTF-16- als auch EBCDIC- und numerische Felder in einer Maske darzustellen. Die COBOL-Anwendung füllt die Felder entsprechend der von IFG erzeugten Adressierungshilfen aus. Die gesamte Maske mit Feldinhalten wird von FHS nach UTFE konvertiert und in den Ausgabepuffer gestellt. openUTM sendet diesen Puffer über VTSU an die Emulation.

In der Transportschicht auf der Client-Seite wird der eintreffende Datenstrom von UTFE nach UTF-8MOD konvertiert. Diese Konvertierung entspricht der Umsetzung von EBCDIC nach ISO8859 bei EBCDIC-Nachrichten. Die Emulation MT9750 überträgt diesen Datenstrom in ihren Bildschirmpuffer und konvertiert ihn zur weiteren Verarbeitung nach UTF-16 (Little Endian).

Die Verarbeitung von Unicode-Daten ist in COBOL2000 ab Version 1.4 mit Hilfe der Klasse national möglich. Ein National-Zeichen in COBOL hat die Codierung UTF-16. Datenfelder der Klasse national können in einer Datenbanktabelle in Spalten mit dem Datentyp NATIONAL CHARACTER (NCHAR) bzw. NATIONAL CHARACTER VARYING (NVCHAR) gespeichert werden.

Detaillierte Informationen dazu entnehmen Sie den Abschnitten „Darstellen und Verarbeiten von Unicode-Zeichen mit COBOL“ auf Seite 37 und „Abspeichern, Suchen und Verwalten von Unicode-Daten in Datenbanken“ auf Seite 39.

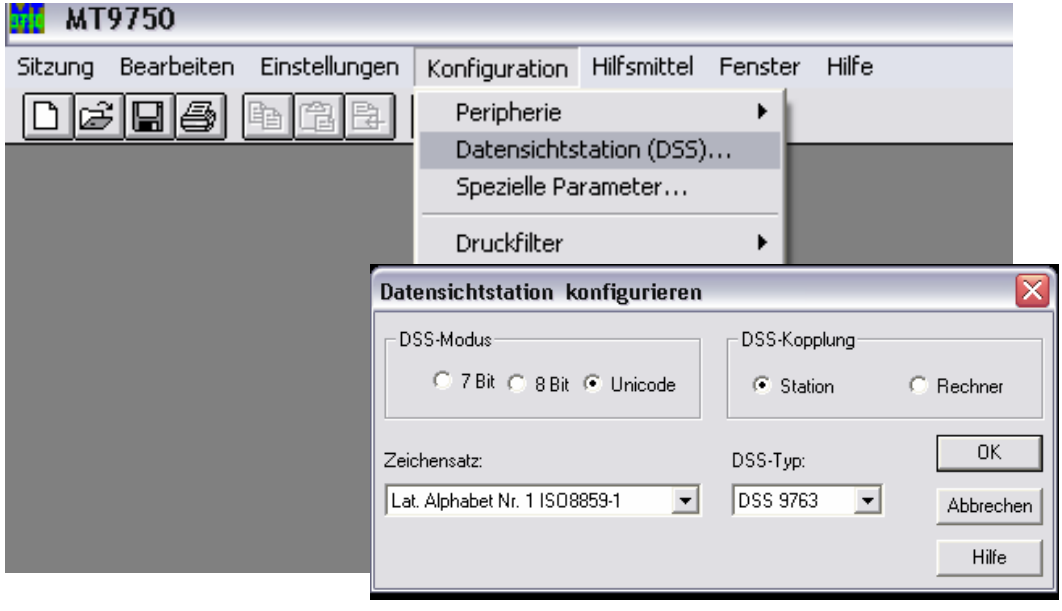
Datenaustausch zwischen unterschiedlichen Systemen

Wenn UTF-16-Zeichenketten von einer Anwendung auf einem System A an eine zweite Anwendung auf einem System B z.B. über die UPIC-Schnittstelle gesendet werden, ist keine Konvertierung notwendig, falls auf beiden Seiten die UTF-16-Codierung gleich ist (Little Endian oder Big Endian). Andernfalls müssen auf einer Seite die beiden Bytes pro UTF-16-Zeichen vertauscht werden.

Die CPI-C-Aufrufe *Convert_Incoming* und *Convert_Outgoing* von UPIC stellen nur eine Konvertierung von EBCDIC nach ASCII bzw. umgekehrt zur Verfügung. Sie können deshalb nicht genutzt werden.

4.3 Konfiguration der Terminalemulation MT9750

Mit der Terminalemulation MT9750 V7.0 ist es möglich, Unicode-Daten an das BS2000/OSD zu senden. Dazu müssen die Sitzungsparameter der Emulation entsprechend konfiguriert werden.



- Wählen Sie im Menü „Konfiguration“ die Option „Datensichtstation (DSS)...“. Das Dialogfeld „Datensichtstation konfigurieren“ wird angezeigt.
- Aktivieren Sie als Datensichtstationsmodus („DSS-Modus“) die Option „Unicode“.
- Wählen Sie als Datensichtstationstyp („DSS-Typ“) die Option „DSS9763“.

Bei Verbindungsaufbau der Emulation mit dem BS2000/OSD-Rechner wird VTSU nun mitgeteilt, dass das Terminal neben den EBCDIC-Zeichensätzen EDF03IRV, EDF041, EDF042 usw. den Zeichensatz UTFE (Unicode) unterstützt. Dies ist Voraussetzung dafür, dass das BS2000-Kommando MODIFY-TERMINAL-OPTIONS CCS = UTFE möglich ist. Die Angabe von UTF-16 ist hier nicht möglich, da der BS2000-Kommandoprozessor kein UTF-16 interpretieren kann.

Mit dem BS2000-Kommando SHOW-TERMINAL-ATTRIBUTES *CAPABILITIES wird die Information, die VTSU von der Emulation erhalten hat, ausgegeben.

Die Nummern hinter dem String CHARACTER-SET-x = sind die so genannten ISO-Nummern. Sie beschreiben welche ISO8859-n-Zeichensätze die Datensichtstation unterstützt. Für UTFE wurde die Pseudo-ISO-Nummer 240 gewählt.

*Beispiel: Ausgabe von /SHOW-TERMINAL-ATTRIBUTES *CAPABILITIES*

```
COLOUR-SUPPORT      = 8           HARDWARE-INFOLINE     = YES
LINE-MODE           = YES         EXTENDED-LINE-MODE    = YES
PHYSICAL-MODE       = YES         FORM-MODE             = YES
PROTOCOL-TYPE       = 810         EXTEND-FIELD-ATTRIB   = YES
STATUS-REQUEST      = YES         ENCRYPTION-SUPPORT    = NO
DOORS-SUPPORT       = NO          DESK2000-SUPPORT      = NO
NUMBER-OF-8-BIT-CHARACTER-SET-SUPPORTED =  7
CHARACTER-SET-1     =  1          CHARACTER-SET-2       =  2
CHARACTER-SET-3     =  5          CHARACTER-SET-4       =  7
CHARACTER-SET-5     =  9          CHARACTER-SET-6       = 15
CHARACTER-SET-7     = 240         CHARACTER-SET-8       = NO
CHARACTER-SET-9     = NO          CHARACTER-SET-10      = NO
CHARACTER-SET-11    = NO          CHARACTER-SET-12      = NO
CHARACTER-SET-13    = NO          CHARACTER-SET-14      = NO
CHARACTER-SET-15    = NO          CHARACTER-SET-16      = NO
```

In diesem Beispiel unterstützt das Terminal die Zeichensätze EDF041, EDF042, EDF045, EDF047, EDF049, EDF04F und UTFE.

Wenn eine Anwendung Unicode-Daten an die Emulation senden oder von ihr empfangen will, muss dies VTSU mitgeteilt werden. Dies kann einerseits über das BS2000-Kommando MODIFY-TERMINAL-OPTIONS, siehe oben, oder andererseits in der Anwendung über den VTSU-B-Kontrollblock erfolgen.

Die Codierung aller Daten, die mit einem Aufruf übergeben werden bzw. empfangen werden, muss in UTFE vorliegen.

Die Einstellung im VTSU-B-Kontrollblock hat Vorrang vor dem mit MODIFY-TERMINAL-OPTIONS eingestellten Wert. Er gilt jedoch nur für den aktuellen Aufruf. Der nächste Aufruf kann mit einem anderen zulässigen Zeichensatz erfolgen. Ein Wechsel des Zeichensatzes hat jedoch ein Löschen des alten Bildschirminhaltes zur Folge.

UTFE ist wie UTF-8 eine variable Codierung der Unicode Code Points. D.h. abhängig vom Code Point werden zur Codierung ein bis maximal fünf Bytes benötigt. Diese Eigenschaft erschwert die Programmierung. Deshalb ist es üblich innerhalb von Programmen mit UCS-2 zu arbeiten. UCS-2 ist der nicht variable Teil von UTF-16, d.h. jedes Zeichen benötigt genau zwei Bytes, siehe auch Abschnitt „Was ist Unicode“ auf Seite 9. Hierbei wird vorausgesetzt, dass der Zeichenvorrat der BMP ausreicht.

Die notwendigen Konvertierungsfunktionen werden von der Systemkomponente XHCS zur Verfügung gestellt, siehe auch Abschnitt „Zeichenbehandlung im BS2000/OSD (XHCS)“ auf Seite 36.

5 Unicode-Anpassungen bei BS2000-Anwendungen

Dieses Kapitel beschreibt die Unicode-Anpassungen in den BS2000-Software-Produkten Programmentwicklung/Test, Datenspeicherung, Drucken und Web-Integration, die der Kunde benötigt, um seine BS2000-Anwendungen an Unicode anzupassen:

- XHCS, die Software zur Zeichenbehandlung im BS2000/OSD
- COBOL-Sprache zur Darstellung und Verarbeitung von Unicode-Zeichen
- AID, die Dialog-Testhilfe
- Datenbanken SESAM/SQL und Oracle 10g für BS2000/OSD
- IFG und FHS zur Erstellung von Formaten und Formatsteuerung
- RSO zur Steuerung der Ausgabe von Druckaufträgen
- Web*Transactions* zur Web-Integration von Unicode-fähigen Anwendungen

Detaillierte Informationen zur Umsetzung der Unicode-Unterstützung finden Sie in den jeweiligen produktspezifischen Handbüchern, siehe auch „Literatur“ auf Seite 99.

5.1 Zeichenbehandlung im BS2000/OSD (XHCS)

E**X**tended **H**ost **C**ode **S**upport (XHCS) verwaltet im BS2000/OSD die Zeichensätze, bietet die Dienste und die Schnittstelle für Codierungsaufgaben und stellt die Konvertierungen zwischen den Codierungen (EBCDIC, Unicode) zur Verfügung.

Die verschiedenen Software-Programme müssen Informationen über Zeichensätze nicht fest speichern, da die XHCS-Schnittstellen jedem Anwenderprogramm zur Verfügung stehen.

Die nachfolgende Liste skizziert die wesentlichen Funktionen von XHCS im Zusammenhang mit der Unicode-Unterstützung. XHCS

- konvertiert von und nach UTF-16, UTF-8 und UTFE sowie von und nach ISO und EBCDIC und bildet die Case-Funktionen (TOUPPER, TOLOWER) ab.
- gibt Informationen über die Code-Kompatibilität aus.
- normalisiert Eingabe-Zeichenketten
- stellt dem Produkt SORT für die Sortierung die erforderlichen Sortiertabellen für die europäischen Sprachen zur Verfügung, siehe auch Abschnitt „Sortierreihenfolge“ auf Seite 24.
- unterstützt die Definition zusätzlicher Unicode-Zeichen. Auch Sortierelemente können Sie sich im BS2000/OSD über XHCS besorgen.

Detaillierte Information zu XHCS entnehmen Sie dem XHCS (BS2000/OSD)-Handbuch.

5.2 Darstellen und Verarbeiten von Unicode-Zeichen mit COBOL

Grundlegende Paradigmen

- COBOL-Programme werden weiterhin in EBCDIC geschrieben.
- Anwendungsdaten werden weiterhin vorwiegend in EBCDIC codiert. Der Unicode-Zeichensatz wird nur verwendet, wo es nötig ist.
- Die einzelne Anwendung muss die Nutzung von Unicode-Daten programmieren.
- Restriktionen bleiben Byte-bezogen gleich.

Umsetzung der UTF-16-Unterstützung

COBOL2000 V1.4 unterstützt UTF-16 durch Implementierung der wesentlichen Sprachmittel zum Datentyp national aus dem COBOL-Standard ISO1989:2002.

Die Unterstützung umfasst:

- PICTURE Maskenzeichen N und USAGE NATIONAL sowie GROUP-USAGE NATIONAL
- nationale Literale und nationale figurative Konstanten
- implizite Konvertierung von EBCDIC nach UTF-16 bei Übertragungen (MOVE) und in Bedingungen
- explizite Konvertierung mit den Funktionen NATIONAL-OF und DISPLAY-OF
- Ersatzzeichen und Ausnahmebehandlung bei Konvertierungen
- Erweiterung der Funktionalität existierender Sprachmittel zur Unterstützung nationaler Daten (insbesondere Bedingungen, INITIALIZE, INSPECT, STRING, UNSTRING, SORT, MERGE und die Funktionen NUMVAL, LOWER-CASE, UPPER-CASE, LENGTH, BYTE-LENGTH und REVERSE).

Daten der Klasse national werden weitgehend analog zu den EBCDIC-Daten der Klasse alphanumerisch definiert und verwendet.

Detaillierte Information zu nationalen Daten und zur Konvertierung zwischen EBCDIC- und UTF-16-Darstellung finden Sie in der COBOL2000 (BS2000/OSD)-Sprachbeschreibung.

5.3 Dialog-Testhilfe AID

Zur Unterstützung von Unicode bietet die Dialog-Testhilfe Advanced Interactive Debugger (AID)

- den Datentyp %UTF16
  Dieser Datentyp entspricht dem Datentyp national, den COBOL2000 im Rahmen seiner Unicode-Unterstützung anbietet, siehe Seite 37. Mit diesem Datentyp hat jedes Zeichen eine Zwei-Bytes-Verschlüsselung.
- Konvertierungsfunktionen von Ein-Byte-EBCDIC nach UTF-16 und umgekehrt: Funktionen %UTF16() und %C()
- die Möglichkeit, Zeichenketten im UTFE-Format einzugeben (U'..')
- die Möglichkeit, die aktuellen Code-Einstellungen und verfügbaren Zeichensätze mit %SHOW %CCSN anzuzeigen

Detaillierte Informationen dazu entnehmen Sie dem AID-Handbuch „Testen von COBOL-Programmen“.

5.4 Abspeichern, Suchen und Verwalten von Unicode-Daten in Datenbanken

Das Abspeichern, Suchen und Verwalten von Unicode-Zeichenketten ermöglichen die Datenbanksysteme SESAM/SQL V5.0 und Oracle 10g für BS2000.

5.4.1 Unicode-Konzept in SESAM/SQL

Das Konzept der Unicode-Unterstützung in SESAM/SQL erlaubt die Verwendung von Unicode-Zeichen in den Spalten von Tabellen und berücksichtigt codierte Zeichensätze in Datenbanken, in Ein-/Ausgabedateien und für Anwenderprogramme.

Die SESAM/SQL-Datenbank enthält im Catalog den Namen eines codierten Zeichensatzes, der angibt, wie Character-Daten der Datenbank interpretiert werden.

Dieses Konzept wirkt sich auf die SQL-Sprachbeschreibung, die Utility-Funktionen und die SESAM/SQL-Anwenderprogramme aus, wie im Folgenden beschrieben.

Grundlegende Informationen zur Unicode-Unterstützung in SESAM/SQL entnehmen Sie dem Basishandbuch von SESAM/SQL-Server (BS2000/OSD).

SQL-Sprachbeschreibung

Neue Datentypen NATIONAL CHARACTER und NATIONAL CHARACTER VARYING

In SESAM/SQL-Datenbanken können Sie Tabellenspalten vom Datentyp NATIONAL CHARACTER und NATIONAL CHARACTER VARYING definieren und darin Unicode-Daten in UTF-16-Format speichern. Metadaten wie beispielsweise Namen von Tabellen, Spalten, Views, werden weiterhin in EBCDIC angegeben.

Die Längen in Bytes der Character-Datentypen werden nicht verändert. Da ein UTF-16-Zeichen zwei Bytes Speicherplatz belegt, ergeben sich für Unicode folgende maximalen Zeichenanzahlen:

NCHAR	128 Zeichen
NVARCHAR	16.000 Zeichen

Unicode-Datentypen werden außerdem unterstützt von den Datenbankfunktionen zur Wiedergewinnung und Änderung von Daten

Außerdem können Host-Variablen und SQL-Literale im UTF-16-Format verwendet werden.

Neue Funktion TRANSLATE

SESAM/SQL bietet über die SQL-Sprache die Konvertierungsfunktion TRANSLATE an zum Umwandeln von EBCDIC-Daten nach Unicode-Daten und umgekehrt (N[VAR]CHAR <–> [VAR]CHAR).

Für die Umsetzung bedient sich SESAM/SQL der von XHCS bereitgestellten Konvertierungsfunktionen.

Utilities

Codierte Zeichensätze werden in folgenden Utility-Funktionen berücksichtigt:

- CREATE CATALOG
  erzeugt eine Datenbank. Beim Erzeugen einer Datenbank kann ein codierter (EBCDIC-)Zeichensatz (Parameter CODE_TABLE) definiert werden.
- ALTER CATALOG
  ermöglicht es, den codierten Zeichensatz einer Datenbank zu ändern.
- EXPORT TABLE
  ermöglicht es, eine Tabelle aus einer Datenbank in eine Exportdatei zu exportieren. Die Exportdatei wird mit dem Coded Character Set Name (CCSN) der Datenbank angelegt.
- IMPORT TABLE
  ermöglicht es, eine Tabelle aus einer Exportdatei in eine Datenbank zu importieren. Der CCSN der Exportdatei wird gegen den CCSN der Datenbank geprüft.
- UNLOAD
  ermöglicht es, Daten einer Tabelle in eine Datei zu entladen. Im Delimiter-Format wird die Datei wahlweise mit dem CCSN der Datenbank oder mit dem CCSN UTFE angelegt. Bei anderen Formaten wird die Datei mit dem CCSN der Datenbank angelegt. Beim Entladen werden die Daten konvertiert, falls nötig.
- LOAD
  ermöglicht es, Daten von einer Datei in eine Tabelle zu laden. Im Delimiter-Format kann auch eine Datei mit dem CCSN UTFE oder einem CCSN vom Typ EBCDIC bearbeitet werden. Ein Datei-CCSN vom Typ EBCDIC wird gegen den CCSN der Datenbank geprüft. Beim Laden werden die Daten konvertiert, falls nötig.

SESAM/SQL-Anwendungen

SESAM/SQL-Anwendungen verbinden sich mit der Datenbank über den sogenannten Database Handler (DBH). Je nachdem, ob der DBH unabhängig (independent) oder eingebunden (linked-in) ist, müssen Sie den Zeichensatz auf unterschiedliche Weise vorgeben. Mit dem neuen Konnektionsmodul-Parameter CCSN (Coded Character Set Name) geben

Sie bei der Anwendung den Zeichensatz an, mit dem die Anwendung Character-Daten interpretiert.
Beim linked-in DBH geschieht dies über eine DBH-Option.

Die SESAM/SQL-Anwendung kann mit der Datenbank zusammenarbeiten, wenn der CCSN der Datenbank und der CCSN der Anwendung identische Werte haben, bzw. wenn für die Datenbank kein codierter Zeichensatz verwendet wird.

Kompatibilität

„Alte“ Datenbanken können ohne Berücksichtigung codierter Zeichensätze unverändert betrieben werden. SESAM kann dann jedoch nicht von und nach UTF-16 konvertieren. Statements, die solche Konvertierungen benötigen, werden abgewiesen.

Ausgabedateien, die Unicode-Daten enthalten, können in älteren SESAM/SQL-Versionen nicht als Eingabedatei verwendet werden.

5.4.2 Unicode-Konzept in Oracle

Oracle 10g unterstützt eine Vielzahl von unterschiedlichen Zeichensätzen, darunter auch mehrere Unicode-Zeichensätze.

Eine Liste aller von Oracle unterstützten Zeichensätze ist im Handbuch "Oracle Database Globalization Support Guide Part Number B14225-02" zu finden.

Der dort aufgeführte Zeichensatz UTFE entspricht dem von IBM definierten Zeichensatz UTF-EBCDIC und nicht dem BS2000-Zeichensatz UTFE.

Die im BS2000/OSD unterstützten Oracle-Zeichensätze sind im Handbuch "Oracle User's Guide for Fujitsu Siemens Computers BS2000/OSD" aufgeführt.

Dabei ist zu unterscheiden zwischen dem Zeichensatz auf Anwendungsseite, dem Zeichensatz für die Datenbank und dem "nationalen" Zeichensatz für Spalten mit Unicode-Datentyp. Oracle konvertiert bei Bedarf mithilfe von eigenen Tabellen zwischen diesen Zeichensätzen, unabhängig von den Zeichensätzen des Betriebssystems.

Als Zeichensatz für die Datenbank und als Zeichensatz auf Anwendungsseite kann in Oracle auf BS2000/OSD kein Unicode-Zeichensatz angegeben werden.

Das Konzept der Unicode-Unterstützung in Oracle auf BS2000/OSD erlaubt jedoch die Verwendung von Unicode-Zeichen in Spalten mit Unicode-Datentyp.

Grundlegende Informationen zur Unicode-Unterstützung in Oracle entnehmen Sie dem Handbuch "Oracle Database Globalization Support Guide Part Number B14225-02".

SQL-Funktionalität

In Oracle-Datenbanken können Sie Unicode-Daten in Tabellenspalten bestimmter Datentypen speichern.

Im Einzelnen handelt es sich um folgende Datentypen:

NCHAR	Textdaten fester Länge, maximal 2000 Bytes
NVARCHAR2	Textdaten variabler Länge, maximal 4000 Bytes
NCLOB	character large object, maximal 4 GB, in UTF-16 gespeichert

Den Zeichensatz von NCHAR- und NVARCHAR2-Daten können Sie beim CREATE DATABASE wahlweise mit UTF-8 oder AL16UTF-16, dem UTF-16-Zeichensatz von Oracle, festlegen.

Die Unicode-Daten in diesen Spalten können Sie über verschiedene Schnittstellen bearbeiten, beispielsweise über SQL und PL/SQL zur Einspeicherung, Wiedergewinnung und Änderung von Daten.

Außerdem können Sie Host-Variablen und SQL-Literale im UTF-16-Format verwenden.

Oracle bietet über SQL die Konvertierungsfunktionen TRANSLATE, TO_CHAR und TO_NCHAR an, zum Umwandeln von EBCDIC-Daten nach Unicode-Daten und umgekehrt.

Außerdem existieren die Funktionen NCHR() und UNISTR() zur Erzeugung von einzelnen Unicode-Zeichen und Unicode-Literalen.

Utilities

Export/Import

Unicode-Daten vom Typ NCHAR, NVARCHAR2 und NCLOB werden beim Export unverändert in eine Binärdatei geschrieben und können damit auch in andere Datenbanken importiert werden.

*SQL*Loader*

Der SQL*Loader ermöglicht es, Unicode-Daten aus einer Eingabedatei in die Datenbank zu laden.

In der Kontrolldatei des SQL*Loader können Sie über den Parameter CHARACTERSET angeben, dass die Daten in der Eingabedatei im UTF-16-Format vorliegen.

Oracle-Anwendungen

Oracle-Anwendungen können mittels der Schnittstellen Pro*Cobol, Pro*C, OCI, JDBC und ODBC Unicode-Daten in der Datenbank bearbeiten.

Kompatibilität

Oracle unterstützt Unicode-Daten in NCHAR Datentypen bereits seit der Version 9i.

In "alten" 8.1.7 Oracle-Datenbanken wurden für Spalten vom Typ NCHAR möglicherweise andere Zeichensätze als Unicode definiert. In diesem Fall ist nach dem Upgrade der Datenbank von 8.1.7 nach 10g zusätzlich ein Upgrade der NCHAR-Spalten erforderlich, wie im Handbuch "Oracle Database Upgrade Guide Part Number B14238-01" beschrieben.

5.5 Unterstützen von Unicode-Feldern in Formaten

Ab Version 8.3A des Format Handling System (FHS) dürfen formatierte Nachrichten zwischen Anwenderprogramm und Terminal auch Unicode-Zeichen enthalten. Dies wird mit Hilfe der sogenannten Unicode-Formate ermöglicht, die mit dem Interaktiven Formatgenerator (IFG) ab Version 8.3 erzeugt werden können.

Unter Unicode-Format ist ein Format zu verstehen, dass mindestens ein Feld enthält, das bei der Formatdefinition in IFG das Attribut UNICODE erhalten hat.

Felder mit dem Attribut UNICODE haben folgende Eigenschaften:

- Der Benutzer kann jedes beliebige Zeichen der BMP in dieses Feld eingeben.
- Der Inhalt des Feldes wird in der Adressierungshilfe des Anwenderprogramms abgelegt. Jedes Zeichen belegt zwei Bytes.
- Die Codierung des Zeichens ist UTF-16.

Terminalemulation und Formate

An der Anwenderschnittstelle unterstützt die Terminalemulation zwei Modi:

- Entweder ist das gesamte Format im Unicode-Modus definiert, d.h. der Anwender kann jedes Unicode-Zeichen in jedes Eingabefeld eingeben.
- Oder das Format ist nicht im Unicode-Modus definiert, d.h. die Eingabe ist auf den eingestellten 7 Bit- oder 8 Bit(ISO8859-x)-Zeichensatz beschränkt.

Eine Mischung dieser beiden Modi in einer Maske unterstützt die Emulation nicht, d.h. die Emulation kann nicht innerhalb einer Maske die Eingabe für ein Feld auf ISO8859-1 beschränken und gleichzeitig für ein anderes Feld in der gleichen Maske die Eingabe von Unicode-Zeichen erlauben.

Prüfungen, die bisher im Modus 2 von der Terminalemulation durchgeführt worden sind, müssen in die Anwendung verlagert werden. Für die sogenannten #-Formate übernimmt FHS diese Prüfungen.

Abarbeitung von #-Formaten in FHS

FHS analysiert bei Unicode-Formaten die UTFE-Eingabezeichenkette des Benutzers, ordnet die Teilzeichenketten den einzelnen Feldern des Formats zu und konvertiert sie entsprechend der Felddefinitionen.

- Wenn dem Feld das Attribut UNICODE zugeordnet ist, konvertiert FHS es nach UTF-16 und überträgt es an die durch die Adressierungshilfe des Formats bestimmte Stelle im Anwenderprogramm.

- Wenn das Feld kein UNICODE-Feld ist, überprüft FHS, ob alle eingegebenen Zeichen mit dem Basiszeichensatz des Formats – festgelegt bei der Generierung in IFG – oder mit dem Zeichensatz, der durch USER/LTERM bei openUTM festgelegt wurde, kompatibel sind. Der aktuell gültige Zeichensatz wird wie bei den 8-Bit Formaten bestimmt. Wenn FHS keinen Fehler erkennt, werden die Zeichen nach EBCDIC konvertiert und an die Stelle im Anwenderprogramm übertragen, die die Adressierungshilfe des Formats festlegt.

  Wenn ein Zeichen nicht in den Zielzeichensatz abgebildet werden kann, löst FHS den ‚field validation check failures' aus, d.h. FHS
  – setzt entweder einen Returncode in die Adressierungshilfe oder
  – gibt eine Standardfehlermeldung aus.

Detaillierte Informationen dazu entnehmen Sie dem Handbuch FHS (BS2000/OSD).

Definition einzelner UNICODE-Felder mit IFG

Das Attribut UNICODE kann in IFG in den Anzeigeeigenschaften der Felder (Maske 0305) zugewiesen werden, sofern das betroffene Feld ein reines Textfeld ist. In den Anzeigeeigenschaften des Formats (Maske 030A) wird die Option „Erfordert UNICODE Unterstützung“ auf „JA“ gesetzt, sobald das Format ein UNICODE-Feld enthält.

In IFG ist es nicht möglich Unicode-Zeichen einzugeben, deshalb können auch keine konstanten Texte oder Auswahlfelder in IFG definiert werden, die solche Zeichen enthalten.

Adressierungshilfen können für Unicode-Formate für die Sprachen COBOL und Assembler mit Hilfe von IFG generiert werden.

Detaillierte Informationen dazu entnehmen Sie dem Handbuch IFG (BS2000/OSD).

5.6 Ausgeben von Druckaufträgen mit Unicode-Daten

Für die Ausgabe von Unicode-Zeichen werden APF-IPDS-Drucker sowie UTF-8-fähige RSO-Drucker unterstützt.

5.6.1 Zentrale Drucker (AFP-IPDS)

Die Druckdatei enthält den Netto-Datenstrom. Eine zweite Datei, die Page Definition, legt fest, wie diese Netto-Daten interpretiert und ausgedruckt werden. Der Wechsel von UNICODE- und EBCDIC-Feldern ist daher möglich.

Wenn eine Textdatei oder bestimmte Zeilen oder Teile von Zeilen in Unicode codiert sind, müssen Sie eine Page Definition anlegen oder entsprechend anpassen und den Unicode-Zeichensatz im Print-Kommando angeben.

5.6.2 Dezentrale Drucker (RSO)

Der Remote Spool Output (RSO) steuert die Ausgabe von Druckaufträgen auf dezentrale Drucker.

Der EBCDIC-basierte Mechanismus von RSO kann auch UTFE-Daten verarbeiten, da der Ein-Byte-Bereich der UTFE-Codierung dem Zeichensatz EBCDIC.DF.03IRV (EDF03IRV) entspricht, siehe auch Abschnitt „UTF-EBCDIC (UTFE)“ auf Seite 17.

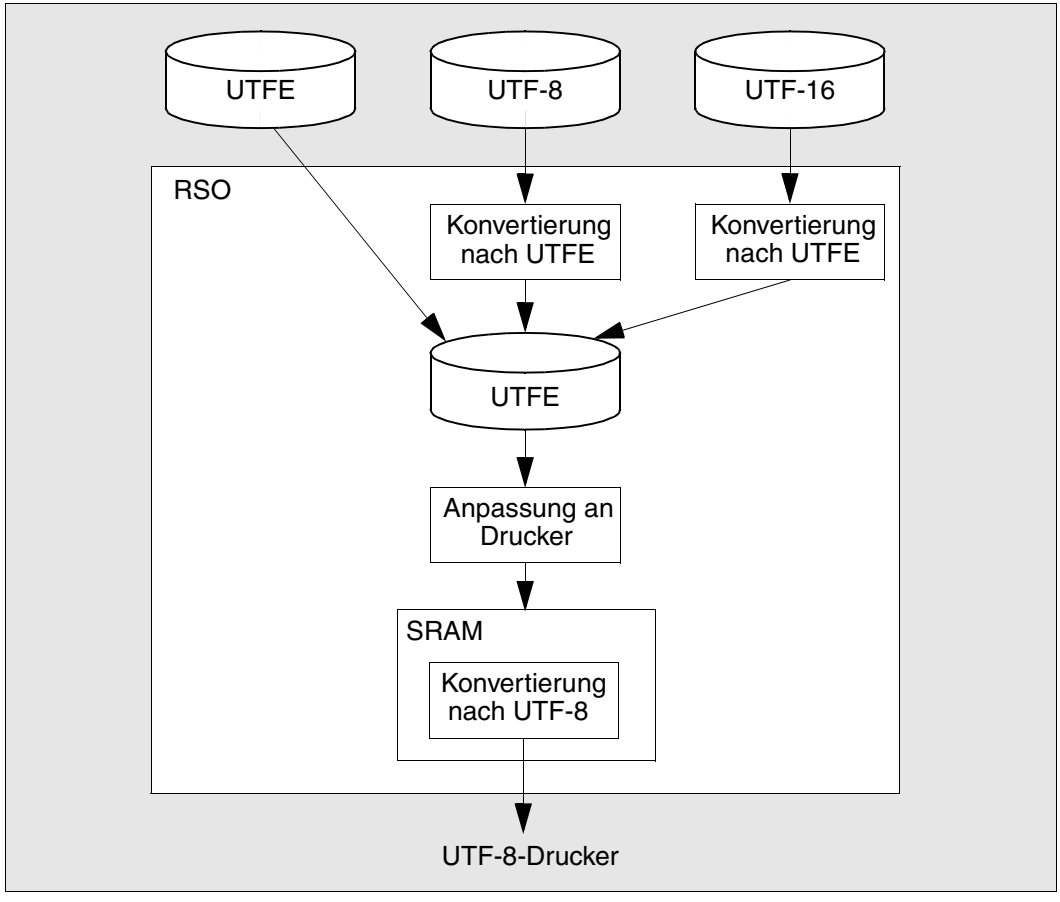


Bild 1: Verarbeitung von Unicode-Dateien

Beim LAN-to-Host-Printing, d.h. beim Ausgeben von Dateien vom PC oder von Unix-basierten Systemen auf RSO-Drucker, müssen UTF-8-Dateien nach UTFE konvertiert werden, bevor sie verarbeitet werden können. Ebenso müssen UTF-16-Daten vor der Verarbeitung nach UTFE konvertiert werden. XHCS liefert die Grundlagen für die Konvertierung von einer Codierung in die andere, siehe auch XHCS (BS2000/OSD)-Handbuch.

Wenn ein Drucker ASCII und UTF-8 unterstützt, wie z.B. der Unicode-fähige UTF-8-Drucker PRINTRONIX P7000, ist RSO in der Lage, sowohl im ASCII-Modus (via EBCDIC) als auch im UTF-8-Modus (via UTFE) zu arbeiten, abhängig von der Codierung der Eingangsdatei. Da RSO den Coded Character Set-Namen der Datei (CCS-NAME) auswertet, ist pro Datei nur ein Zeichensatz möglich.

Detaillierte Informationen zu RSO entnehmen Sie dem Handbuch „RSO (BS2000/OSD)“.

5.7 Web-Integration von Unicode-fähigen Anwendungen (Web*Transactions*)

Im Rahmen der Unicode-Unterstützung erzeugt Web*Transactions* UTF-8-codierte Daten. Diese Daten werden zum Browser und wieder zurück durchgeleitet, um Unicode-fähige Host-Anwendungen zu unterstützen.

Die Templates informieren den Browser, welche Zeichen-Codierung er zum Anzeigen der Daten und zum Versenden der Antwort verwenden soll,
– entweder über das Attribut charset im HTTP-Header Content-Type oder
– über das HTML-Tag
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />.

Biz*Transactions* unterstützt Unicode nicht.

Web*Transactions* for OSD

Im Rahmen der Unicode-Unterstützung durch das BS2000/OSD wird das von Web*Transactions* for OSD unterstütze Terminal-Protokoll 9750 Unicode-fähig.

Die Unicode-Unterstützung muss explizit eingeschaltet werden, andernfalls verhält sich Web*Transactions* for OSD kompatibel zu den Vorversionen.

Dazu wurde ein neuer Terminal-Typ eingeführt:
Das Systemobjekt-Attribut TERMINAL_TYPE kennt nun auch den Wert 9763-UNICODE.

Wenn Sie diesen Wert einstellen, gibt sich die in den Host-Adapter integrierte Emulation beim Verbindungsaufbau zum BS2000/OSD als Unicode-fähig zu erkennen. Die jeweilige BS2000/OSD-Anwendung hat dann die Möglichkeit, diese Fähigkeit zu nutzen.

Wenn die Host-Anwendung in den Unicode-Modus umschaltet, werden

- die Feldinhalte vom/zum BS2000/OSD als UTF-EBCDIC interpretiert,
- die entsprechenden Inhalte der Hostobjekte bei der Auswertung UTF-8-codiert ausgegeben, sowie bei der Zuweisung als UTF-8-codiert interpretiert.

Der Host-Adapter zeigt dies im Systemobjekt-Attribut HOST_CHARSET durch den neuen Wert UTF-8 an.

Da alle von Web*Transactions* für diesen Host-Adapter erzeugten Templates das Systemobjekt-Attribut HOST_CHARSET auf das globale Systemobjekt-Attribut CHARSET zuweisen, wird der Content-Type im HTTP-Header entsprechend gesetzt, sodass der Browser die Daten korrekt interpretieren kann.

Detaillierte Informationen dazu entnehmen Sie dem Web*Transactions*-Handbuch „Anschluss an OSD-Anwendungen“.

Web*Transactions* for openUTM

Im Rahmen der Unicode-Unterstützung durch das BS2000/OSD kann der von Web*Transactions* for openUTM verwendete Host-Adapter Daten an der UPIC-Schnittstelle auch als Unicode-Zeichen verarbeiten.

Der zum Formulieren der Bildschirmmasken verwendete Interaktive Formatgenerator (IFG) ermöglicht es, einzelnen Feldern das Attribut UNICODE zuzuweisen (siehe auch Seite 44). Mit dem OSD-Programm IFG2FLD werden die Format-Beschreibungen in ein für Web*Lab* lesbares Format – die Formatbeschreibungsquelle – exportiert.

IFG2FLD berechnet die Offsets der einzelnen Felder im UPIC-Puffer abhängig vom Attribut UNICODE. Diese Formatbeschreibungsquelle wird von dem in Web*Lab* enthaltenen Generator unter Berücksichtigung des Unicode-Kennzeichens in Format-spezifische FLD-Dateien konvertiert. Die FLD-Dateien müssen für alle Formate, die Unicode-Felder enthalten, neu generiert werden. Früher generierte Templates können unverändert bestehen bleiben, falls die Umstellung auf Unicode-Felder die einzige Änderung in dem Format war. Ggf. müssen Sie an zentraler Stelle oder in diesen Format-spezifischen Templates die Zuweisung des Wertes UTF-8 auf das Systemobjekt-Attribut CHARSET einfügen.

Dieses neue Attribut Unicode steht auch den Web*Transactions*-Anwendungen als Hostobjekt-Attribut zur Verfügung: Es kann die Werte ‚Y' und ‚N' annehmen.

- Für alle Felder, die mit Unicode=='Y' markiert sind, werden die Host-Daten im UPIC-Puffer als UTF-16-Daten betrachtet und in UTF-8-Daten zum Browser umgesetzt, bzw. die UTF-8-Browser-Daten werden zu UTF-16-Host-Daten umgesetzt.
- Die ASCII-EBCDIC-Konvertierung, die abhängig vom Systemobjekt-Attribut HOST_CHAR_CODE ist, wird nur bei Unicode=='N' durchgeführt.

Zusätzlich wird über das gleichnamige Attribut an dem Host-Steuerobjekt WT_HOST_MESSAGE.Unicode angezeigt, ob in der aktiven Nachricht mindestens eine Feld mit Unicode=='Y' enthalten ist.
Abhängig von diesem Attribut kann das Template das globale Systemobjekt-Attribut CHARSET auf den Wert UTF-8 setzen. Die von Web*Transactions* generierten Templates enthalten diese Zuweisung automatisch. Bei bereits bestehenden Templates, die nicht neu generiert werden sollen, müssen Sie dies ggf. manuell einfügen.

Detaillierte Informationen dazu entnehmen Sie dem Web*Transactions*-Handbuch „Anschluss an openUTM-Anwendungen über UPIC".

6 Unicode-Anpassungen bei der Dateiverarbeitung

Dieses Kapitel gibt einen Überblick über die Ausgestaltung der Unicode-Unterstützung in folgenden Dateiverarbeitungsprogrammen des BS2000/OSD:

- EDT zur Aufbereitung von Unicode-Dateien
- PERCON zur Konvertierung und Normalisierung
- SORT zur Feldsortierung
- *open*FT zur Dateiübertragung

Detaillierte Informationen zur Umsetzung der Unicode-Unterstützung finden Sie in den jeweiligen produktspezifischen Handbüchern, siehe auch „Literatur“ auf Seite 99.

6.1 Erstellen und Editieren von Unicode-Dateien (EDT)

Der Editor zur Dateiaufbereitung (EDT) ermöglicht es, Unicode-Dateien in den Zeichensätzen UTF-8, UTF-16 oder UTFE zu erstellen und zu editieren.

Der EDT V17.0 kann in einem V16.6-kompatiblen Kompatibilitäts-Modus und in einem Unicode-Modus betrieben werden.

- Im Unicode-Modus kann EDT V17.0A in Unicode codierte Dateien bearbeiten.

  Dem Anwender, der in Unicode codierte Dateien bearbeiten will, wird eine komfortable Unterstützung geboten: Dazu zählen u.a. die Möglichkeit, unterschiedlich codierte Dateien in verschiedenen Arbeitsdateien des EDT gleichzeitig zu bearbeiten sowie die Aufhebung der Begrenzung der Satzlänge (bisher 256 Zeichen). Der EDT kann beim Schreiben in eine Datei maximal 32.768 Byte lange Sätze verarbeiten.

  Die Unicode-Darstellung in den Arbeitsdateien hat zur Konsequenz, dass alle Schnittstellen, an denen der Anwender direkten Zugriff auf die Sätze der Arbeitsdatei hat, nicht kompatibel bleiben können. Dies trifft auf die L-Modus Unterprogramm-Schnittstelle, auf die @RUN-Schnittstelle und auf den Locate-Mode der IEDTGLE-Schnittstelle zu. Diese Schnittstellen können daher nicht mehr verwendet werden, wenn man die neuen Funktionen nutzen will.

- Der Kompatibilitäts-Modus bietet die volle Funktionalität des EDT V16.6B inklusive der alten L-Modus-Unterprogramm-Schnittstelle. Eine neue @MODE-Anweisung im Kompatibilitätsmodus ermöglicht das Umschalten in den Unicode-Modus.

Detaillierte Informationen dazu entnehmen Sie dem Handbuch EDT (BS2000/OSD).

6.2 Konvertieren und Normalisieren von Unicode-Dateien (PERCON)

Der Peripheral Converter (PERCON) bietet die Möglichkeit, SAM-Dateien oder Teile davon, die nicht in Unicode codiert sind, in Unicode-Format umzusetzen und umgekehrt. Konvertiert wird immer dann, wenn der Eingabe- und der Ausgabedatei unterschiedliche Coded Character Set Names zugeordnet sind.

Da Unicode-Daten in nicht-normalisierter Form vorliegen können, bietet PERCON darüber hinaus die Möglichkeit, diese Daten in die zusammengesetzte Zeichendarstellung umzuwandeln, siehe auch Abschnitt „Normalisierung“ auf Seite 23.

Detaillierte Informationen dazu entnehmen Sie dem Handbuch PERCON V2.9A (BS2000/OSD).

6.3 Sortieren von Unicode-Feldern (SORT)

SORT bietet eine Sortiermöglichkeit für Felder, die Unicode-Zeichen enthalten, nach der Unicode Default Collation Table. Diese Tabelle enthält die Werte für die Sortierelemente, siehe auch Web-Seite des Unicode-Konsortiums
http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt.
Sortieren ist derzeit für den Unicode-Zeichensatz UTF-16 möglich, bei dem jedes Zeichen durch zwei Bytes repräsentiert wird (ohne Surrogate Pairs).

Jedem UTF-16-Zeichen wird ein sogenanntes Sortierelement zugeteilt, das die Reihenfolge festlegt, nach der die UTF-16-Zeichen sortiert werden, siehe auch Abschnitt „Sortierreihenfolge“ auf Seite 24. Die Sortierelemente selbst werden mittels einer von XHCS gelieferten Tabelle festgelegt, siehe auch XHCS (BS2000/OSD)-Handbuch. Diese Tabelle enthält eine Wertigkeit des Zeichens auf verschiedenen Ebenen. Nähere Informationen zu den verschiedenen Ebenen entnehmen Sie dem Abschnitt „Sortierreihenfolge“ auf Seite 24.

Detaillierte Informationen zur Funktionsweise von SORT entnehmen Sie dem Handbuch SORT V7.9A (BS2000/OSD).

6.4 Übertragen von Unicode-Dateien (*open*FT)

Wenn Sie Dateien mit *open*FT-Partnern ab V10 übertragen, können Sie die lokal und remote für die Datenkonvertierung zu verwendenden codierten Zeichensätze im Auftrag zuweisen. Mit diesen Partnersystemen können auch Unicode-Dateien übertragen werden.

Die zu verwendenden codierten Zeichensätze legen Sie entweder über eine Parameterangabe im TRANSFER-FILE-Kommando fest oder bei DMS-Dateien im Dateikatalog.

Bei der Dateiübertragung innerhalb des BS2000/OSD wird für die Code-Konvertierung XHCS genutzt.
Für die Dateiübertragung in andere Systeme sind die Standard-Code-Tabellen für die Konvertierung in *open*FT implementiert und können über Betriebsparameter in *open*FT eingestellt werden.

Detaillierte Informationen zum File Transfer mit *open*FT entnehmen Sie den Handbüchern zu openFT V10.0A für BS2000/OSD.

7 Tipps und Tricks

7.1 Tipps zu SESAM/SQL

Wenn Sie mit SESAM/SQL Unicode-Dateien bearbeiten, ist Folgendes zu beachten:

CCSN der Datenbank / CCSN der Anwendung

Mit Hilfe der Anweisung ALTER CATALOG ALTER CODE_TABLE können Sie den CCSN (Code-Table) der Datenbank ändern. Dabei ist zu beachten, dass eine Anwendung nur dann korrekt über den DBH auf eine Datenbank zugreifen kann, wenn der Parameter CCSN in der Konfigurationsdatei der Anwendung mit dem CCSN der Datenbank übereinstimmt. Bei einer Anwendung mit linked-in-DBH muss der Parameter CODED-CHARACTER-SET der DBH-Option LINKED-IN-ATTRIBUTES identisch sein mit dem CCSN der Datenbank.

Falls die Werte nicht identisch sind, ist der Zugriff auf die Datenbank nur möglich, wenn für die Datenbank der CCSN namens _NONE_ definiert ist.

Weiter ist der Zugriff auf die Datenbank mit Hilfe des Utility Monitors auf die Metadaten der Datenbank über die SNF- und INF-Masken immer möglich. In der SNF.1 Maske wird der aktuell für die Datenbank definierte CCSN ausgegeben. In der CNF-Maske wird der in der Konfigurationsdatei der Anwendung (bzw. DBH-Optionsdatei bei linked-in-DBH) eingetragene CCSN ausgegeben. Mit Hilfe dieser beiden Masken können Sie überprüfen, ob die CCSN übereinstimmen.

CCSN des Bildschirms

In der SQL-Maske des Utility Monitors werden Sätze einer Treffermenge eines SELECT Statements ausgegeben.

Alphanumerische Werte vom Datentyp CHARACTER (VARYING) werden so übernommen wie sie in der Datenbank stehen, während numerische und Zeitdatentypen nach CHARACTER konvertiert werden. NATIONAL Werte vom Datentyp N[VAR]CHAR werden gemäß CCSN der Datenbank nach CHARACTER konvertiert. Für nicht konvertierbare Werte wird das Ersatzkennzeichen „Punkt (.)“ ausgegeben.

Am Bildschirm werden die Zeichen in dem CCSN dargestellt, der für den Bildschirm eingestellt ist. Die Einstellung kann mit dem BS2000 Kommando SHOW-TERMINAL-OPTIONS überprüft werden und bei Bedarf mit dem Kommando MODIFY-TERMINAL-OPTIONS geändert werden.

Achtung: dieses Kommando ist nur zulässig solange noch kein Programm geladen ist.

Wenn der CCSN, der für den Bildschirm eingestellt ist, vom CCSN der Datenbank abweicht, dann werden manche Zeichen am Bildschirm nicht korrekt dargestellt.

Unload im Delimiter-Format mit UTFE

Beim Unload im Delimiter-Format mit UTFE werden die Werte der Spalten nach UTFE konvertiert und in die Unload-Datei ausgegeben. Dabei ist zu beachten, dass die Codierung in UTFE zwischen einem und fünf byte lang sein kann.

7.2 Tipps zu LMS

Bei der Verwendung vom LMS und Unicode-Dateien ist Folgendes zu beachten:

EXTRACT-ELEMENT in ISAM Datei

LMS extrahiert Bibliothekselemente standardmäßig als ISAM-Datei mit einem 8 byte langen EBCDIC ISAM-Schlüssel. Dieser kann von EDT als Zeilennummer interpretiert werden. Handelt es sich um ein Element mit ISO (ASCII) Zeichensatz , so erzeugt LMS eine Datei, die sowohl EBCDIC- als auch ASCII-Zeichen enthält. Diese Datei kann dann unter Umständen mit anderen Produkten, wie z.B. EDT, nicht mehr vernünftig bearbeitet werden. Noch problematischer wird es bei UTF16. Dann besteht der ISAM-Schlüssel aus Ein-Byte-Zeichen und die Daten aus Zwei-Byte-Zeichen.

Der Anwender kann jedoch durch Angabe von ACCESS-METHOD = *SAM die Elemente extrahieren oder eine vom EDT erzeugte ISAM-Datei mit //ADD-ELEMENT in eine Bibliothek aufnehmen. Dabei wird standardmäßig der ISAM-Schlüssel mit Warnung entfernt (auch bei 16 byte langem Schlüssel).

Man kann aber auch mit SOURCE-ATTRIBUTES = *KEEP den Schlüssel im Element speichern lassen und später das Element unverändert wieder mit Schlüssel extrahieren und mit dem EDT V17 weiter bearbeiten.

EDIT-ELEMENT

Mit der LMS-Anweisung EDIT-ELEMENT können Elemente mittels EDT bearbeitet werden. LMS nutzt dabei die EDT Unterprogrammschnittstelle im Kompatibilitätsmodus (EDT-UP V16). Elemente in Unicode-Zeichensatz sind mit EDIT-ELEMENT nicht bearbeitbar, da die Schnittstelle EDT-UP V16 kein Unicode beherrscht. LMS übergibt an der EDT-UP-Schnittstelle die Anweisung @CODENAME UTF8/16/E. Diese wird von EDT V17 mit einem Returncode abgelehnt.

Der Anwender kann das Element jedoch aus dem EDT heraus mit @OPEN LIBRARY=… bearbeiten.

ADD-ELEMENT und EXTRACT-ELEMENT aus EDT Anweisungszeile (@USE)

Mit Hilfe der EDT-Anweisung @USE können bestimmte LMS-Anweisungen in der EDT-Anweisungszeile ausgeführt werden. Dazu gehören die Anweisungen ADD-ELEMENT und EXTRACT-ELEMENT.

- ADD-ELEMENT (lesen aus EDT-Arbeitsdatei)

  Der Anwender hat in einer EDT-Arbeitsdatei Unicode-Daten erzeugt. Danach will er diese Daten mit ADD-ELEMENT aus der EDT-Anweisungszeile in ein Bibliothekselement schreiben. EDT übergibt die Anweisung ADD-ELEMENT an LMS und LMS versucht nun, die Daten mittels der EDT-UP-Funktion IEDTGET zu lesen. EDT lehnt diesen Vorgang mit einem Returncode ab, weil LMS die Daten über die EDT-UP V16 Schnittstelle lesen will.

- EXTRACT-ELEMENT (schreiben in EDT-Arbeitsdatei)

  Der Anwender will aus der EDT-Anweisungszeile mittels der LMS-Anweisung EXTRACT-ELEMENT ein Unicode-Element in eine EDT-Arbeitsdatei schreiben. Diesen Vorgang lehnt EDT wie bei der EDIT-ELEMEN ab, weil auch hier LMS versucht, eine @CODENAME UTF8/16/E-Anweisung über die V16 UP- Schnittstelle an EDT zu übergeben.

Unicode-Elemente können jedoch mit den EDT-Funktionen @WRITE LIBRARY=… und @COPY LIBRARY=… in eine Bibliothek geschrieben bzw. aus ihr gelesen werden.

8 Anhang

8.1 Unicode-Produkte im BS2000/OSD: Überblick und Abhängigkeiten

BS2000/OSD-BC ab der Version 6.0B ist Voraussetzung für alle Unicode-Produkte.

Produkt	**Version**	**setzt voraus (zusätzlich zu BS2000/OSD-BC ab 6.0B)**
BS2000/OSD-BC	6.0B (2. Korrekturpaket 2006) enthält – CRTE-BASYS 1.6 – CRTE-MSG 1.6 – SPOOL 4.8 – SPSERVE 2.9 – SYSFILE 15.0B	–
	7.0 enthält – CRTE-BASYS 1.6 – CRTE-MSG 1.6 – SPOOL 4.8 – SPSERVE 2.9 – SYSFILE 16.0	–
AID	3.2	*open*Net Server 3.2, MT9750 7.0
COBOL2000	1.4	XHCS 2.0, CRTE 2.6
CRTE	2.6	–
CRTE-BASYS	1.6	–
CRTE-MSG	1.6	–
EDT	17.0	*open*Net Server 3.2
ESQL-COBOL (COBOL SQL-Precompiler)	3.0	SESAM 5.0, COBOL2000 1.4

Tabelle 5: Unicode-Produkte: Überblick und Abhängigkeiten

Produkt	**Version**	**setzt voraus (zusätzlich zu BS2000/OSD-BC ab 6.0B)**
FHS	8.3	*open*Net Server 3.2, MT9750 7.0, IFG 8.3
IFG	8.3	FHS 8.3, COBOL2000 1.4
MT9750	7.0	*open*Net Server 3.2
*open*FT	10.0	–
*open*Net Server	3.2 enthält – XHCS 2.0 – VTSU-B 13.2	–
ORACLE	10g	–
PERCON	2.9	XHCS 2.0
RSO	3.5	XHCS 2.0
SESAM/SQL-Server	5.0	COBOL2000 1.4, ESQL-COBOL 3.0, XHCS 2.0
SORT	7.9	XHCS 2.0
SYSFILE	15.0B für OSD 6.0B	–
	16.0 für OSD 7.0	–
VTSU-B	13.2	MT9750 7.0
Web*Transactions*	7.1	*open*Net Server 3.2
XHCS	2.0	–

Tabelle 5: Unicode-Produkte: Überblick und Abhängigkeiten

8.2 Erweiterte BS2000-Makros

GCCSN – CCS-Namen für Kommando- und Dateneingabe anzeigen

Neu ist der Parameter STREAM zur Ausgabe des Coded Character Set Names für die BS2000-Systemdateien.

GCCSN
STREAM = SYSDTA/SYSCMD/SYSOUT/SYSLST/SYSLST(1)..(99) , ...

STREAM=
Name der Systemdatei, für die der CCS-Name ausgegeben werden soll.

Hinweis

Wenn SYSOUT/SYSLST einer Datei oder einem Bibliothekselement zugewiesen wurde, kann sich der CCS-Namen zwischen dem Öffnen und dem Schließen der Datei bzw. dem Bibliothekselement nicht ändern.
Wenn jedoch SYSOUT einem Terminal zugewiesen ist, kann der Benutzer den CCS-Namen dynamisch in seinem Programm über den VTSUC-B für den aktuellen WROUT-Auftrag temporär ändern. Diese Änderung gilt nur für den aktuellen Auftrag und kann deshalb über GCCSN-Makro nicht ermittelt werden.

WROUT – Satz nach SYSOUT übertragen

Neu ist der Parameter ASSIGN für die Angabe, ob Änderungen der SYSOUT-Zuweisung angezeigt werden sollen.

WROUT
satz,fehler, , ... , [ASSIGN = NO / YES]

ASSIGN=
legt fest, ob Änderungen der SYSOUT-Zuweisung angezeigt werden sollen. Das Benutzerprogramm wird von der anfänglichen Standardzuweisung und von jeder folgenden Zuweisung für SYSOUT über die Fehleradresse benachrichtigt.
Der Schreibvorgang erfolgt nicht bei einer erkannten Änderung.
Dieser Operand ist nur für eine 31-Bit-Schnittstelle zulässig.

NO
Änderungen der SYSOUT-Zuweisung werden nicht angezeigt.

YES
Änderungen der SYSOUT-Zuweisung werden angezeigt.

Hinweis

Die SYSOUT-Zuweisung wird in einem Ausgabefeld der Parameterliste angezeigt und wird von der SYSFILE-Verarbeitung versorgt.

Dadurch hat das Benutzerprogramm die Möglichkeit, auf die geänderten Randbedingungen zu reagieren und eventuell notwendige Umsetzungen der Ausgabezeichenkette durchführen zu können, um dann den Schreibvorgang mit der korrigierten Ausgabezeichenkette erneut anzustoßen.

TSTAT – Datenstationseigenschaften abfragen / DCSTA – Operandentabelle für Datenstationseigenschaften generieren

Mit dem Makro TSTAT fordern Sie im Teilnehmerbetrieb Informationen über die Datenstation an. DCSTA generiert Empfangsfelder oder symbolische Feldnamen (DSECTs) für die Informationen, die Sie durch den Aufruf von TSTAT erhalten.

Im Abschnitt BASIC der DSECT DCSTA im Feld STACSSx wird für Unicode-fähige Terminaltypen der Wert x'F0' ausgegeben.

VTSUCB – VTSU-Parameter für Ein-/Ausgabe erstellen

Wenn Unicode-Felder am Terminal ausgegeben werden sollen, müssen diese nach UTFE konvertiert werden.

Der CCS-Name in VTSUCB muss dann mit UTFE versorgt sein. Andere Unicode-CCS-Namen (UTF16 oder UTF8) sind nicht zulässig.

8.3 Coded Character Set Names (CCSN): Standardwerte

Variable/Kommando/ Programm	**Standardwert**	**Bemerkung**
CLASS-2-Option HOSTCODE	EDF03IRV	
ADD-USER	aus CLASS-2-Option HOSTCODE	
MODIFY-USER-ATTR CCS=*STD	aus CLASS-2-Option HOSTCODE	
VTSU	TIAM/UTM/DCAM-PTERM8=N: – EDF03IRV TIAM/UTM/DCAM-PTERM8=Y: – 8-BIT-Terminal: CCSN des HOME-PUB-SET – 7-BIT-TERMINAL: CCSN=EDF03IRV	Die Parameter TIAM-PTERM8, UTM-PTERM8 und DCAM-PTERM8 stehen in der VTSU-Parameterdatei unter TSOS. Sie gelten systemweit.
MODIFY-TERMINAL-OPTIONS CCS=*8-BIT-DEFAULT	8-BIT-TERMINAL: – CCSN= CCSN des HOME-PUB-SET 7-BIT-TERMINAL: – Fehlermeldung	Sie können einen CCSN auch explizit angeben. Dieser muss jedoch vom Terminal unterstützt werden.
EDT	Nach dem Laden des EDT entspricht – im DIALOG der CCSN dem in VTSU eingestellten CCSN – im BATCH der CCSN dem CCSN der zugewiesenen Anweisungseingabedatei	EDT im Unicode-Modus: Jedes Fenster kann Daten mit einem eigenen CCSN haben. Wenn Daten von einem Fenster in ein anderes kopiert werden, werden die Daten konvertiert. EDT im Kompatibilitätsmodus: Im Dialog kann sich der CCSN durch das Einlesen einer Datei auf den der Datei ändern, sofern noch keine Daten in einem der EDT-Fenster vorhanden ist. D.h. Alle EDT-Fenster können nur Daten mit dem gleichen CCSN enthalten.

Tabelle 6: Standardwerte für CCSN

Variable/Kommando/ Programm	Standardwert	Bemerkung
BS2000/OSD: Anlegen einer Datei: CREATE-FILE CREATE-FILE-GROUP MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES	CCSN des aufnehmenden Public Volume Set (PVS) = EDF03IRV -> CCSN der Datei = *NONE CCSN des aufnehmenden PVS != EDF03IRV -> Die Datei erhält den CCSN des Benutzereintrags des PVS.	
LMS EDIT-ELEMENT ELEMENT=*NONE TO-ELEMENT=<output>	Der EDT bestimmt den CCSN, siehe EDT.	
PLAM	CCS=*NONE CCS= CCS der PLAM-Bibliothek	

Tabelle 6: Standardwerte für CCSN

Aktion	Kommando	Standardwert	Bemerkung
Festlegen des Systemstandard-zeichensatz	CLASS-2-Option HOSTCODE=<ccsn>	EDF03IRV	Auswertung erfolgt bei Kommando ADD-USER, bei XHCS CCSN=*SYSDEF
Einrichten einer Benutzerkennung	ADD-USER	Aus CLASS-2-Option HOSTCODE	Legt den CCS-Namen im Benutzereintrag des HOME-Public Volume Set (PVS) fest. Auswertung: XHCS CCSN=*USRDEF VTSU-Makro DCSTA STACURCH
Einrichten eines Benutzereintrags zu einer Benutzerkennung	ADD-USER	Aus CLASS-2-Option HOSTCODE	Legt den CCS-Namen im Benutzereintrag des PVS fest.

Tabelle 7: Aktionen und zugehörige Standardwerte für CCSN

8.4 Hilfreiche Code-Tabellen

8.4.1 Konvertierung von ISO8859 nach EBCDIC (BS2000/OSD) und umgekehrt

ISO8859-n und EBCDIC.DF.04.n haben den gleichen Zeichenvorrat. Konvertiert von dem einen Zeichensatz in den anderen wird mit folgenden Umsetzungstabellen:

Konvertierung von ISO8859 nach EBCDIC (BS2000/OSD)

	-0	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**	**-8**	**-9**	**-A**	**-B**	**-C**	**-D**	**-E**	**-F**
0-	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
1-	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
2-	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3-	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4-	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5-	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BB	BC	BD	6A	6D
6-	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7-	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	FB	4F	FD	FF	07
8-	20	21	22	23	24	04	06	08	28	29	2A	2B	2C	09	0A	14
9-	30	31	25	33	34	35	36	17	38	39	3A	3B	1A	1B	3E	5F
A-	41	AA	B0	B1	9F	B2	D0	B5	79	B4	9A	8A	BA	CA	AF	A1
B-	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
C-	64	65	62	66	63	67	9E	68	74	71	72	73	78	75	76	77
D-	AC	69	ED	EE	EB	EF	EC	BF	80	E0	FE	DD	FC	AD	AE	59
E-	44	45	42	46	43	47	9C	48	54	51	52	53	58	55	56	57
F-	8C	49	CD	CE	CB	CF	CC	E1	70	C0	DE	DB	DC	8D	8E	DF

Tabelle 8: Konvertierung von ISO8859 nach EBCDIC (BS2000/OSD)

Konvertierung von EBCDIC (BS2000) nach ISO8859

	-0	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**	**-8**	**-9**	**-A**	**-B**	**-C**	**-D**	**-E**	**-F**
0-	00	01	02	03	85	09	86	7F	87	8D	8E	0B	0C	0D	0E	0F
1-	10	11	12	13	8F	0A	08	97	18	19	9C	9D	1C	1D	1E	1F
2-	80	81	82	83	84	92	17	1B	88	89	8A	8B	8C	05	06	07
3-	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
4-	20	A0	E2	E4	E0	E1	E3	E5	E7	F1	60	2E	3C	28	2B	7C
5-	26	E9	EA	EB	E8	ED	EE	EF	EC	DF	21	24	2A	29	3B	9F
6-	2D	2F	C2	C4	C0	C1	C3	C5	C7	D1	5E	2C	25	5F	3E	3F
7-	F8	C9	CA	CB	C8	CD	CE	CF	CC	A8	3A	23	40	27	3D	22
8-	D8	61	62	63	64	65	66	67	68	69	AB	BB	F0	FD	FE	B1
9-	B0	6A	6B	6C	6D	6E	6F	70	71	72	AA	BA	E6	B8	C6	A4
A-	B5	AF	73	74	75	76	77	78	79	7A	A1	BF	D0	DD	DE	AE
B-	A2	A3	A5	B7	A9	A7	B6	BC	BD	BE	AC	5B	5C	5D	B4	D7
C-	F9	41	42	43	44	45	46	47	48	49	AD	F4	F6	F2	F3	F5
D-	A6	4A	4B	4C	4D	4E	4F	50	51	52	B9	FB	FC	DB	FA	FF
E-	D9	F7	53	54	55	56	57	58	59	5A	B2	D4	D6	D2	D3	D5
F-	30	31	32	33	34	35	36	37	38	39	B3	7B	DC	7D	DA	7E

Tabelle 9: Konvertierung von EBCDIC (BS2000) nach ISO8859

Nicht belegte Byte-Positionen in ISO8859-/ EBCDIC-Tabellen

Tabelle	**Nicht belegte ISO8859-Position**	**Nicht belegte EBCDIC.DF.04-Position**
ISO8859-1	-	-
ISO8859-2	-	-
ISO8859-3	A5,AE,BE,C2,D0,E3,F0	B2,AF,B9,62,AC,46,8C
ISO8859-4	-	-
ISO8859-5	-	-
ISO8859-7	AE,D2	AF,ED
ISO8859-15	-	-

Tabelle 10: Nicht belegte Byte-Positionen in ISO8859-/ EBCDIC -Tabellen

Inhalt von ISO- und entsprechenden EBCDIC-Code-Tabellen

ISO-Tabelle	EBCDIC BS2000/OSD	Inhalt
ISO/IEC 8859-1	EBCDIC.DF.04-1	Part 1: Latin alphabet No. 1
ISO/IEC 8859-2	EBCDIC.DF.04-2	Part 2: Latin alphabet No. 2
ISO/IEC 8859-3	EBCDIC.DF.04-3	Part 3: Latin alphabet No. 3
ISO/IEC 8859-4	EBCDIC.DF.04-4	Part 4: Latin alphabet No. 4
ISO/IEC 8859-5	EBCDIC.DF.04-5	Part 5: Latin/Cyrillic alphabet
ISO/IEC 8859-6		Part 6: Latin/Arabic alphabet
ISO/IEC 8859-7	EBCDIC.DF.04-7	Part 7: Latin/Greek alphabet
ISO/IEC 8859-8		Part 8: Latin/Hebrew alphabet
ISO/IEC 8859-9	EBCDIC.DF.04-9	Part 9: Latin alphabet No. 5
ISO/IEC 8859-10		Part 10: Latin alphabet No. 6
ISO/IEC 8859-11		Part 11: Latin/Thai alphabet
ISO/IEC 8859-13		Part 13: Latin alphabet No. 7
ISO/IEC 8859-14		Part 14: Latin alphabet No. 8 (Celtic)
ISO/IEC 8859-15	EBCDIC.DF.04-F	Part 15: Latin alphabet No. 9
ISO/IEC 8859-16		Part 16: Latin alphabet No. 10

Tabelle 11: Inhalt von ISO- und entsprechenden EBCDIC-Code-Tabellen

8.4.2 Nach ISO8859.n konvertierbare Unicode-Zeichen

Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Alle in der folgenden Tabelle aufgelisteten Unicode-7 Bit-Zeichen können in folgende Zeichensätze konvertiert werden:

– ISO8859.1 bzw. EBCDIC.DF.04-1
– ISO8859.2 bzw. EBCDIC.DF.04-2
– ISO8859.3 bzw. EBCDIC.DF.04-3
– ISO8859.4 bzw. EBCDIC.DF.04-4
– ISO8859.5 bzw. EBCDIC.DF.04-5
– ISO8859.7 bzw. EBCDIC.DF.04-7
– ISO8859.9 bzw. EBCDIC.DF.04-9
– ISO8859.15 bzw. EBCDIC.DF.04-15

Unicode Code Point	Zeichen
u+0000	0
u+0001	START OF HEADING
u+0002	START OF TEXT
u+0003	END OF TEXT
u+0004	END OF TRANSMISSION
u+0005	ENQUIRY
u+0006	ACKNOWLEDGE
u+0007	BELL
u+0008	BACKSPACE
u+0009	HORIZONTAL TABULATION
u+000A	LINE FEED
u+000B	VERTICAL TABULATION
u+000C	FORM FEED
u+000D	CARRIAGE RETURN
u+000E	SHIFT OUT
u+000F	SHIFT IN
u+0010	DATA LINK ESCAPE
u+0011	DEVICE CONTROL ONE
u+0012	DEVICE CONTROL TWO
u+0013	DEVICE CONTROL THREE
u+0014	DEVICE CONTROL FOUR
u+0015	NEGATIVE ACKNOWLEDGE
u+0016	SYNCHRONOUS IDLE
u+0017	END OF TRANSMISSION BLOCK
u+0018	CANCEL
u+0019	END OF MEDIUM
u+001A	SUBSTITUTE
u+001B	ESCAPE
u+001C	FILE SEPARATOR
u+001D	GROUP SEPARATOR

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Unicode Code Point	Zeichen
u+001E	RECORD SEPARATOR
u+001F	UNIT SEPARATOR
u+0020	SPACE
u+0021	!
u+0022	"
u+0023	#
u+0024	$
u+0025	%
u+0026	&
u+0027	'
u+0028	(
u+0029	)
u+002A	*
u+002B	"+"
u+002C	,
u+002D	-
u+002E	.
u+002F	/
u+0030	0
u+0031	1
u+0032	2
u+0033	3
u+0034	4
u+0035	5
u+0036	6
u+0037	7
u+0038	8
u+0039	9
u+003A	:
u+003B	;

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Unicode Code Point	Zeichen
u+003C	<
u+003D	=
u+003E	>
u+003F	?
u+0040	@
u+0041	A
u+0042	B
u+0043	C
u+0044	D
u+0045	E
u+0046	F
u+0047	G
u+0048	H
u+0049	I
u+004A	J
u+004B	K
u+004C	L
u+004D	M
u+004E	N
u+004F	O
u+0050	P
u+0051	Q
u+0052	R
u+0053	S
u+0054	T
u+0055	U
u+0056	V
u+0057	W
u+0058	X
u+0059	Y

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Unicode Code Point	Zeichen
u+005A	Z
u+005B	[
u+005C	\
u+005D	]
u+005E	^
u+005F	_
u+0060	`
u+0061	a
u+0062	b
u+0063	c
u+0064	d
u+0065	e
u+0066	f
u+0067	g
u+0068	h
u+0069	i
u+006A	j
u+006B	k
u+006C	l
u+006D	m
u+006E	n
u+006F	o
u+0070	p
u+0071	q
u+0072	r
u+0073	s
u+0074	t
u+0075	u
u+0076	v
u+0077	w

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Unicode Code Point	Zeichen
u+0078	x
u+0079	y
u+007A	z
u+007B	{
u+007C	|
u+007D	}
u+007E	~
u+007F	<control> DELETE
u+0080	<control>
u+0081	<control>
u+0082	<control>= BREAK PERMITTED HERE
u+0083	<control>= NO BREAK HERE
u+0084	<control>
u+0085	<control>=NEXT LINE (NL)
u+0086	<control>= START OF SELECTED AREA
u+0087	<control>= END OF SELECTED AREA
u+0088	<control>= CHARACTER TABULATION SET
u+0089	<control>= CHARACTER TABULATION WITH JUSTIFICATION
u+008A	<control>= LINE TABULATION SET
u+008B	<control>= PARTIAL LINE FORWARD
u+008C	<control>= PARTIAL LINE BACKWARD
u+008D	<control>= REVERSE LINE FEED
u+008E	<control>= SINGLE SHIFT TWO
u+008F	<control>= SINGLE SHIFT THREE
u+0090	<control>= DEVICE CONTROL STRING
u+0091	<control>= PRIVATE USE ONE
u+0092	<control>= PRIVATE USE TWO
u+0093	<control>= SET TRANSMIT STATE
u+0094	<control>= CANCEL CHARACTER

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Unicode Code Point	Zeichen
u+0095	<control>= MESSAGE WAITING
u+0096	<control>= START OF GUARDED AREA
u+0097	<control>= END OF GUARDED AREA
u+0098	<control>= START OF STRING
u+0099	<control>
u+009A	<control>= SINGLE CHARACTER INTRODUCER
u+009B	<control>= CONTROL SEQUENCE INTRODUCER
u+009C	<control>= STRING TERMINATOR
u+009D	<control>=OPERATING SYSTEM COMMAND
u+009E	<control>= PRIVACY MESSAGE
u+009F	<control>= APPLICATION PROGRAM COMMAND

Tabelle 12: Nach ISO8859-n konvertierbare Unicode-7 Bit-Zeichen

Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+00A0	1,2,3,4,5,7,9,15	NO-BREAK SPACE
u+00A1	1,9,15	¡
u+00A2	1,9,15	¢
u+00A3	1,3,7,9,15	£
u+00A4	1,2,3,5,9	¤
u+00A5	1,9,15	¥
u+00A6	1,7,9	¦
u+00A7	1,2,3,4,5,7,9,15	§
u+00A8	1,2,3,4,7,9	¨
u+00A9	1,7,9,15	©
u+00AA	1,9,15	ª
u+00AB	1,7,9,15	«
u+00AC	1,7,9,15	¬
u+00AD	1,2,3,4,5,7,9,15	-
u+00AE	1,9,15	®
u+00AF	1,4,9,15	¯
u+00B0	1,2,3,4,7,9,15	°
u+00B1	1,7,9,15	±
u+00B2	1,3,7,9,15	²
u+00B3	1,3,7,9,15	³
u+00B4	1,2,3,4,9	´
u+00B5	1,3,9,15	µ
u+00B6	1,9,15	¶
u+00B7	1,3,7,9,15	·
u+00B8	1,2,3,9	¸
u+00B9	1,9,15	¹
u+00BA	1,9,15	º
u+00BB	1,7,9,15	»
u+00BD	1,3,7,9	½
u+00BE	1,9	¾
u+00BF	1,9,15	¿

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+00C0	1,3,9,15	À
u+00C1	1,2,3,4,9,15	Á
u+00C2	1,2,3,4,9,15	Â
u+00C3	1,4,9,15	Ã
u+00C4	1,2,3,4,9,15	Ä
u+00C5	1,4,9,15	Å
u+00C6	1,4,9,15	Æ
u+00C7	1,2,3,9,15	Ç
u+00C8	1,3,9,15	È
u+00C9	1,2,3,4,9,15	É
u+00CA	1,3,9,15	Ê
u+00CB	1,2,3,4,9,15	Ë
u+00CC	1,3,9,15	Ì
u+00CD	1,2,3,4,9,15	Í
u+00CE	1,2,3,4,9,15	Î
u+00CF	1,3,9,15	Ï
u+00D0	1,15	Đ
u+00D1	1,3,9,15	Ñ
u+00D2	1,3,9,15	Ò
u+00D3	1,2,3,9,15	Ó
u+00D4	1,2,3,4,9,15	Ô
u+00D5	1,4.9,15	Õ
u+00D6	1,2,3,4,9,15	Ö
u+00D7	1,2,3,4,9,15	×
u+00D8	1,4,9,15	Ø
u+00D9	1,3,9,15	Ù
u+00DA	1,2,3,4,9,15	Ú
u+00DB	1,3,4,9,15	Û
u+00DC	1,2,3,4,9,15	Ü
u+00DD	1,2,15	Ý
u+00DE	1,15	Þ

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+00DF	1,2,3,4,9,15	ß
u+00E0	1,3,9,15	à
u+00E1	1,2,3,4,9,15	á
u+00E2	1,2,3,4,9,15	â
u+00E3	1,4,9,15	ã
u+00E4	1,2,3,4,9,15	ä
u+00E5	1,4,9,15	å
u+00E6	1,4,9,15	æ
u+00E7	1,2,3,9,15	ç
u+00E8	1,3,9,15	è
u+00E9	1,2,3,4,9,15	é
u+00EA	1,3,9,15	ê
u+00EB	1,2,3,4,9,15	ë
u+00EC	1,3,9,15	ì
u+00ED	1,2,3,4,9,15	í
u+00EE	1,2,3,4,9,15	î
u+00EF	1,3,9,15	ï
u+00F0	1,15	ð
u+00F1	1,3,9,15	ñ
u+00F2	1,3,9,15	ò
u+00F3	1,2,3,9,15	ó
u+00F4	1,2,3,4,9,15	ô
u+00F5	1,4,9,15	õ
u+00F6	1,2,3,4,9,15	ö
u+00F7	1,2,3,4,9,15	÷
u+00F8	1,4,9,15	ø
u+00F9	1,3,9,15	ù
u+00FA	1,2,3,4,9,15	ú
u+00FB	1,3,4,9,15	û
u+00FC	1,2,3,4,9,15	ü
u+00FD	1,2,15	ý

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+00FE	1,15	þ
u+00FF	1,9,15	ÿ
u+0100	4	Ā
u+0101	4	ā
u+0102	2	Ă
u+0103	2	ă
u+0104	2,4	Ą
u+0105	2,4	ą
u+0106	2	Ć
u+0107	2	ć
u+0108	3	Ĉ
u+0109	3	ĉ
u+010A	3	Ċ
u+010B	3	ċ
u+010C	2,4	Č
u+010D	2,3	č
u+010E	2	Ď
u+010F	2	ď
u+0110	2,4	Đ
u+0111	2	đ
u+0112	4	Ē
u+0113	4	ē
u+0116	4	Ė
u+0117	4	ė
u+0118	2,4	Ę
u+0119	2,4	ę
u+011A	2	Ě
u+011B	2	ě
u+011C	3	Ĝ
u+011D	3	ĝ
u+011E	3,9	Ğ

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+011F	3,9	ğ
u+0120	3	Ġ
u+0121	3	ġ
u+0122	4	Ģ
u+0123	4	ģ
u+0124	3	Ĥ
u+0125	3	ĥ
u+0126	3	Ħ
u+0127	3	ħ
u+0128	4	Ĩ
u+0129	4	ĩ
u+012A	4	Ī
u+012B	4	ī
u+012E	4	Į
u+012F	4	į
u+0130	3,9	İ
u+0131	3,9	ı
u+0134	3	Ĵ
u+0135	3	ĵ
u+0136	4	Ķ
u+0137	4	ķ
u+0138	4	ĸ
u+0139	2	Ĺ
u+013A	2	ĺ
u+013B	4	Ļ
u+013C	4	ļ
u+013D	2	Ľ
u+013E	2	ľ
u+0141	2	Ł
u+0142	2	ł
u+0143	2	Ń

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+0144	2	ń
u+0145	4	Ņ
u+0146	4	ņ
u+0147	2	Ň
u+0148	2	ň
u+014A	4	Ŋ
u+014B	4	ŋ
u+014C	4	Ō
u+014D	4	ō
u+0150	2	Ő
u+0151	2	ő
u+0152	15	Œ
u+0153	15	œ
u+0154	2	Ŕ
u+0155	2	ŕ
u+0156	4	Ŗ
u+0157	4	ŗ
u+0158	2	Ř
u+0159	2	ř
u+015A	2	Ś
u+015B	2	ś
u+015C	3	Ŝ
u+015D	3	ŝ
u+015E	2,3,9	Ş
u+015F	2,3,9	ş
u+0160	2,4,15	Š
u+0161	2,4,15	š
u+0162	2	Ţ
u+0163	2	ţ
u+0164	2	Ť
u+0165	2	ť

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+0166	4	Ŧ
u+0167	4	ŧ
u+0168	4	Ũ
u+0169	4	ũ
u+016A	4	Ū
u+016B	4	ū
u+016C	3	Ŭ
u+016D	3	ŭ
u+016E	2	Ů
u+016F	2	ů
u+0170	2	Ű
u+0171	2	ű
u+0172	4	Ų
u+0173	4	ų
u+0178	15	Ÿ
u+0179	2	Ź
u+017A	2	ź
u+017B	2,3	Ż
u+017C	2,3	ż
u+017D	2,4,15	Ž
u+017E	2,4,15	ž
u+02C7	2,4	ˇ
u+02D8	2,3	˘
u+02D9	2,3,4	˙
u+02DB	2,4	˛
u+02DD	2	˝
u+037A	7	ͺ
u+0384	7	΄
u+0385	7	΅
u+0386	7	Ά
u+0388	7	Έ

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+0389	7	Ή
u+038A	7	Ί
u+038C	7	Ό
u+038E	7	Ύ
u+038F	7	Ώ
u+0390	7	ΐ
u+0391	7	Α
u+0392	7	Β
u+0393	7	Γ
u+0394	7	Δ
u+0395	7	Ε
u+0396	7	Ζ
u+0397	7	Η
u+0398	7	Θ
u+0399	7	Ι
u+039A	7	Κ
u+039B	7	Λ
u+039C	7	Μ
u+039D	7	Ν
u+039E	7	Ξ
u+039F	7	Ο
u+03A0	7	Π
u+03A1	7	Ρ
u+03A3	7	Σ
u+03A4	7	Τ
u+03A5	7	Υ
u+03A6	7	Φ
u+03A7	7	Χ
u+03A8	7	Ψ
u+03A9	7	Ω
u+03AA	7	Ϊ

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+03AB	7	Ϋ
u+03AC	7	ά
u+03AD	7	έ
u+03AE	7	ή
u+03AF	7	ί
u+03B0	7	ΰ
u+03B1	7	α
u+03B2	7	β
u+03B3	7	γ
u+03B4	7	δ
u+03B5	7	ε
u+03B6	7	ζ
u+03B7	7	η
u+03B8	7	θ
u+03B9	7	ι
u+03BA	7	κ
u+03BB	7	λ
u+03BC	7	μ
u+03BD	7	ν
u+03BE	7	ξ
u+03BF	7	ο
u+03C0	7	π
u+03C1	7	ρ
u+03C2	7	ς
u+03C3	7	σ
u+03C4	7	τ
u+03C5	7	υ
u+03C6	7	φ
u+03C7	7	χ
u+03C8	7	ψ
u+03C9	7	ω

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+03CA	7	ϊ
u+03CB	7	ϋ
u+03CC	7	ό
u+03CD	7	ύ
u+03CE	7	ώ
u+0401	5	Ё
u+0402	5	Ђ
u+0403	5	Ѓ
u+0404	5	Є
u+0405	5	S
u+0406	5	I
u+0407	5	Ї
u+0408	5	J
u+0409	5	Љ
u+040A	5	Њ
u+040B	5	Ћ
u+040C	5	Ќ
u+040E	5	Ў
u+040F	5	Џ
u+0410	5	А
u+0411	5	Б
u+0412	5	В
u+0413	5	Г
u+0414	5	Д
u+0415	5	Е
u+0416	5	Ж
u+0417	5	З
u+0418	5	И
u+0419	5	Й
u+041A	5	К
u+041B	5	Л

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+041C	5	М
u+041D	5	Н
u+041E	5	О
u+041F	5	П
u+0420	5	Р
u+0421	5	С
u+0422	5	Т
u+0423	5	У
u+0424	5	Ф
u+0425	5	Х
u+0426	5	Ц
u+0427	5	Ч
u+0428	5	Ш
u+0429	5	Щ
u+042A	5	Ъ
u+042B	5	Ы
u+042C	5	Ь
u+042D	5	Э
u+042E	5	Ю
u+042F	5	Я
u+0430	5	а
u+0431	5	б
u+0432	5	в
u+0433	5	г
u+0434	5	д
u+0435	5	е
u+0436	5	ж
u+0437	5	з
u+0438	5	и
u+0439	5	й
u+043A	5	к

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+043B	5	л
u+043C	5	м
u+043D	5	н
u+043E	5	о
u+043F	5	п
u+0440	5	р
u+0441	5	с
u+0442	5	т
u+0443	5	у
u+0444	5	ф
u+0445	5	х
u+0446	5	ц
u+0447	5	ч
u+0448	5	ш
u+0449	5	щ
u+044A	5	ъ
u+044B	5	ы
u+044C	5	ь
u+044D	5	э
u+044E	5	ю
u+044F	5	я
u+0451	5	ё
u+0452	5	ђ
u+0453	5	ѓ
u+0454	5	є
u+0455	5	s
u+0456	5	і
u+0457	5	ї
u+0458	5	ј
u+0459	5	љ
u+045A	5	њ

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Unicode Code Point	ISO8859-n / EBCDIC.DF.04.n	Zeichen
u+045B	5	ћ
u+045C	5	ќ
u+045E	5	ў
u+045F	5	џ
u+2015	7	–
u+2018	7	‘

Tabelle 13: Nach ISO8859-n konvertierbare Unicode-Zeichen

Zeichen des deutschen Meldewesens ohne Entsprechung in ISO8859-n

Unicode Code Point	Zeichen
u+0114	Ĕ
u+0115	ĕ
u+012C	Ĭ
u+012D	ĭ
u+014E	Ŏ
u+014F	ŏ
u+0166	Ŧ
u+0167	ŧ
u+0174	Ŵ
u+0175	ŵ
u+0176	Ŷ
u+0177	ŷ
u+019D	Ɲ
u+01A0	Ơ
u+01A1	ơ
u+01AF	Ư
u+01B0	ư
u+01CD	Ǎ
u+01CE	ǎ
u+01CF	Ǐ
u+01D0	ǐ
u+01D1	Ǒ
u+01D2	ǒ
u+01D3	Ǔ
u+01D4	ǔ
u+01E6	Ǧ
u+01E7	ǧ

Tabelle 14: Zeichen des deutschen Meldewesens ohne Entsprechung in ISO8859-n

Unicode Code Point	Zeichen
u+01F4	Ǵ
u+01F5	ǵ
u+0212	R̂
u+0213	r̂
u+0272	ɲ
u+1E20	Ḡ
u+1E21	ḡ
u+1E24	Ḥ
u+1E25	ḥ
u+1E30	Ḱ
u+1E31	ḱ
u+1E44	Ṅ
u+1E45	ṅ
u+1E60	Ṡ
u+1E61	ṡ
u+1E62	Ṣ
u+1E63	ṣ
u+1E84	Ẅ
u+1E85	ẅ
u+1E8E	Ẏ
u+1E8F	ẏ
u+1E90	Ẑ
u+1E91	ẑ
u+1E92	Ẓ
u+1E93	ẓ
u+1EA0	Ạ
u+1EA1	ạ

Tabelle 14: Zeichen des deutschen Meldewesens ohne Entsprechung in ISO8859-n

Unicode Code Point	Zeichen
u+1EAA	Ẫ
u+1EAB	ẫ
u+1EBC	Ẽ
u+1EBD	ẽ
u+1EC4	Ễ
u+1EC5	ễ
u+1ECA	Ị
u+1ECB	ị
u+1ECC	Ọ
u+1ECD	ọ
u+1ED6	Ỗ
u+1ED7	ỗ
u+1EE4	Ụ
u+1EE5	ụ
u+1EF2	Ỳ
u+1EF3	ỳ
u+1EF8	Ỹ
u+1EF9	ỹ
u+2264	≤

Tabelle 14: Zeichen des deutschen Meldewesens ohne Entsprechung in ISO8859-n

Fachwörter

Basic Multilanguage Plane
: entspricht UCS-2
UCS-2 kann nur den BMP-Zeichenvorrat darstellen, aber Zeichen aus der BMP können in UCS-2, UTF-8, usw. gespeichert werden.

Big Endian
: Art der Bytes-Anordnung durch den Prozessor bei einer bestimmten Codierung.
Bei Big Endian liegt das höchstwertige Byte an der niedrigsten Speicheradresse.

Byte Order Marks
: Die Byte Order Marks geben an, ob ein UTF-16 bzw. UTF-32 String in Little Endian- oder Big Endian-Format vorliegt.

Code Point
: siehe Unicode Code Point

Code Space
: Menge aller Code Points. Der Code Space der Unicode-Norm V4 kennt 1.114.112 Code Points. Unterteilt ist der Code Space in Ebenen, die jeweils 65.536 Code Points umfassen.

Collation Element
: Sortierelement
Die Unicode-Norm beschreibt einen linguistischen Sortieralgorithmus, der darauf basiert, dass jedem Zeichen ein Sortierelement zugeordnet wird. Es besteht aus einer Folge von bis zu drei Ebenen.

diakritisches Zeichen
: mit einem Grundzeichen oder Symbol verknüpftes Zeichen, z. B. Akzent, Tilde.

Little Endian

Art der Bytes-Anordnung durch den Prozessor bei einer bestimmten Codierung.
Bei Little Endian liegt das niedrigstwertige Byte an der niedrigsten Speicheradresse.

Normalisierung

Vorgang zur einheitlichen Darstellung von Zeichen, die mehrere Unicode Code Points haben können (z.B. diakritische Zeichen).

Surrogate Pairs

Alle Zeichen oberhalb von U+FFFF der Unicode-Codierung UTF-16. Diese werden durch 4 Bytes dargestellt.
U+D800 - U+DBFF: High Surrogate
U+DC00- U+DFFF: Low Surrogate
Ein Paar High und Low Surrogate bilden die Code Points von U+10000 bis U+10FFFF ab.

Unicode Code Point

In Unicode wird jedem Zeichen eine Nummer, der so genannte Code Point, zugeordnet.
Ein Unicode Code Point wird im Allgemeinen in der Form U+n angegeben, wobei n aus 4 bis 6 hexadezimalen Ziffern besteht.

Universal Character Set der Länge 2 (UCS-2)

Bei UTF-16 werden alle Unicode-Zeichen zwischen U+0000 und U+FFFF durch 2 Bytes codiert. Die Zeichen in diesem Bereich werden auch als Universal Character Set der Länge 2 (UCS-2) bezeichnet. Der Bereich der Surrogates U+D800 bis U+DFFF, sowie die Byte Order Marks U+FEFF und U+FFFE sind ausgenommen.

UTF-8

Unicode-Codierung, definiert vom Unicode-Konsortium.
UTF-8 verwendet eine variable Anzahl von Bytes zur Codierung der Unicode-Zeichen. Die Byte-Darstellung der ASCII-Zeichen bleibt unverändert. Bei einem Zeichen, das mit mehreren Bytes codiert ist, repräsentiert keines der Einzelbytes ein gültiges Zeichen.

UTF-8MOD (modifiziertes UTF-8)

Vorstufe zu UTF-EBCDIC. Die Ein-Byte-Codierung von UTF-8 wird um den zweiten Steuerzeichenblock (U+80 - 8+9F) erweitert. Die Anzahl der möglichen Folge-Bytes reduziert sich entsprechend. Bei einer anschließenden Umsetzung von UTF-8MOD nach EBCDIC bleibt die Darstellung der EDF03IRV-Zeichen unverändert.
Ein Technischer Report des Unicode-Konsortiums schlägt für Systeme, die EBCDIC verwenden, eine Konvertierung von Unicode-Zeichen in EBCDIC vor.

UTF-16

Unicode-Codierung, definiert vom Unicode-Konsortium.
Bei UTF-16 sind alle Unicode-Zeichen zwischen U+0000 und U+FFFF durch 2 Bytes codiert. Alle Zeichen oberhalb von U+FFFF werden durch 4 Bytes dargestellt, so genannte Surrogate Pairs.

UTF-32

Unicode-Codierung, definiert vom Unicode-Konsortium.
Jedes Zeichen des Unicode-Standards wird direkt als 32-Bits-Einheit codiert.

UTF-EBCDIC (UTFE)

Unicode-Codierung für Maschinen, die den EBCDIC-Zeichensatz verwenden

Abkürzungen

BMP	Basic Multilanguage Plane
ISO	International Standardization Organisation
UCS-2	Universal Character Set 2
UCS-4	Universal Character Set 4
UTF-8	UCS Transformation Format 8 Bit oder Unicode Transformation Format
UTF-8MOD	modifiziertes UTF-8
UTF-16	UCS Transformation Format 16 Bit
UTFE	im BS2000/OSD: UTF-EBCDIC

Tabellen

Literatur

Die Handbücher sind online unter *http://manuals.fujitsu-siemens.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://FSC-manualshop.com* zu bestellen.

[1] **XHCS** (BS2000/OSD)
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

[2] **PERCON V2.9A** (BS2000/OSD)
Benutzerhandbuch

[3] **COBOL2000** (BS2000/OSD)
COBOL-Compiler
Sprachbeschreibung

[4] **AID** (BS2000/OSD)
Advanced Interactive Debugger
Basishandbuch

[5] **AID** (BS2000/OSD)
Testen von COBOL-Programmen

[6] **IFG** (BS2000/OSD)
Interaktiver Formatgenerator
Benutzerhandbuch

[7] **FHS** (BS2000/OSD)
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

[8] **SESAM/SQL-Server** (BS2000/OSD)
Basishandbuch
Benutzerhandbuch

[9] **SORT V7.9A** (BS2000/OSD)
Benutzerhandbuch

[10] **EDT** (BS2000/OSD)
Anweisungen
Benutzerhandbuch

[11] *open*FT V10.0A für BS2000/OSD
Enterprise File Transfer in der offenen Welt
Benutzerhandbuch

[12] *open*FT V10.0A für BS2000/OSD
Installation und Administration
Sysemverwalterhandbuch

[13] **RSO** (BS2000/OSD)
Remote Spool Output

[14] Web*Transactions*
Konzepte und Funktionen

[15] Web*Transactions*
Anschluss an OSD-Anwendungen
Benutzerhandbuch

[16] Web*Transactions*
Anschluss an openUTM-Anwendungen über UPIC
Benutzerhandbuch

Zusätzliche Literatur

Internetseite des UNICODE-Konsortiums:
http://www.unicode.org/standard/translations/german.html

Oracle User's Guide for Fujitsu Siemens Computers BS2000/OSD
http://www.oracle.com/technology/documentation/database10gr2.html

Oracle Database Globalization Support Guide
Part Number B14225-02

Oracle Database Upgrade Guide
Part Number B14238-01

Stichwörter

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.